

PENGEMBANGAN WEB *SERVICE* PADA IOT DATA *STORAGE* DENGAN PENDEKATAN SEMANTIK DAN PENAMBAHAN MEKANISME AUTENTIKASI

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Tuti Wardani Hamid
NIM: 145150201111155



PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

PENGEMBANGAN WEB SERVICE PADA IOT DATA STORAGE DENGAN
PENDEKATAN SEMANTIK DAN PENAMBAHAN MEKANISME AUTENTIKASI

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Tuti Wardani Hamid
NIM: 145150201111155

Skripsi ini telah diuji dan dinyatakan lulus pada
2 Agustus 2018

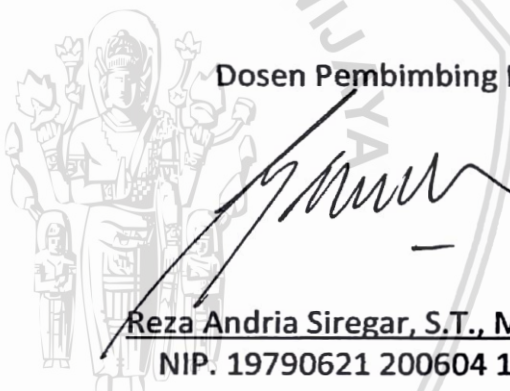
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I



Eko Sakti P., S.Kom., M.Kom.
NIK. 201102 860805 1 001

Dosen Pembimbing II



Reza Andria Siregar, S.T., M.Kom.
NIP. 19790621 200604 1 003

Mengetahui

Ketua Jurusan Teknik Informatika



Tri Astoto Kurniawan, S.T, M.T, Ph.D

NIP: 19710518 200312 1 001

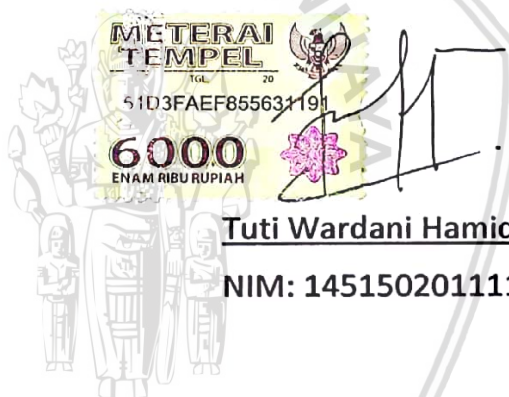


PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata di dalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 2 Agustus 2018



Tuti Wardani Hamid

NIM: 145150201111155

KATA PENGANTAR

Assalamualaikum warahmatullahi wabarakatuh.

Puji syukur penulis panjatkan kehadiran Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga laporan skripsi yang berjudul “PENGEMBANGAN WEB SERVICE PADA IOT DATA STORAGE DENGAN PENDEKATAN SEMANTIK DAN PENAMBAHAN MEKANISME AUTENTIKASI “ dapat terselesaikan.

Selama pengerjaan laporan skripsi penulis mendapat banyak pelajaran dan pengalaman baru. Penulis juga banyak mengaplikasikan ilmu yang didapat selama menduduki bangku perkuliahan ke dalam laporan skripsi ini. Penulis menyadari bahwa laporan skripsi ini tidak akan berhasil tanpa dukungan, doa, dan bantuan dari beberapa pihak. Oleh karena itu, penulis ingin menyampaikan rasa erima kasih dan rasa hormat kepada :

1. Bapak Eko Sakti Pramukantoro, S.Kom.,M.Kom., dan bapak Reza Andria Siregar, S.T.,M.Kom., selaku dosen pembimbing skripsi penulis yang begitu sabar membimbing penulis juga memberikan arahan dengan sangat baik kepada penulis sehingga dapat menyelesaikan skripsi ini.
2. Ibu dan keluarga besar saya yang tidak henti-hentinya memberikan doa dan dukungan selama proses menempuh pendidikan di Malang termasuk saat proses pengerjaan skripsi ini.
3. Bapak Wayan Firdaus Mahmudy, S.Si.,M.T., Ph.D., selaku Dekan Fakultas Ilmu Komputer
4. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D., selaku Ketua Jurusan Teknik Informatika
5. Bapak Agus Wahyu Widodo, S.T., M.Cs., selaku Ketua Program Studi Teknik Informatika
6. Maxi Luckies G, Binariyanto Aji, Hilman Nihri, Uis Yudha Tri Wirawan, Landika Hari Suganda, Ahmad Naufal, Jessy Ratna W, yang selalu meluangkan waktu untuk membantu dan berbagi ilmu sehingga laporan skripsi ini dapat terselesaikan
7. Nadya Ramadana, Shindy Maria Ulfa, Khairinnisa Rifna, Meylisa Dwiwati Marali, Nirmala Faizah Saraswati, dan seluruh anggota keluarga L-Tifam yang selalu memberikan bantuan, saran, dan semangat mulai awal perkuliahan di Malang sampai dengan proses pengerjaan skripsi ini selesai
8. Keluarga 13 yang selalu memberikan semangat, dukungan dan doa selama proses pengerjaan skripsi

9. Keluarga Besar Advokesma Eksekutif Mahasiswa Universitas Brawijaya 2014 dan 2015, yang telah memberikan banyak pengalaman dan dukungan moral
10. Keluarga UKM INKAI Universitas Brawijaya yang selalu berbagi ilmu dan memberikan semangat, doa, dan banyak pengalaman menyenangkan
11. Keluarga Advokesma Badan Eksekutif Mahasiswa Fakultas Ilmu Komputer 2016 dan 2017, yang telah memberikan banyak pengalaman, semangat, doa dan dukungan moral
12. Keluarga Besar Badan Eksekutif Mahasiswa Fakultas Ilmu Komputer Kabinet Berbeda dan Kabinet Sukma Karya yang berbagi cerita dan pengalaman yang menyenangkan selama proses penyusunan skripsi
13. Seluruh civitas akademika Fakultas Ilmu Komputer yang selalu memberikan dukungan dan bantuan dalam proses penyelesaian skripsi ini
14. Teman-teman keminatan KBJ dan teman – teman Teknik Informatika Angkatan 2014 lainnya yang selalu berbagi ilmu dari awal perkuliahan sampai tahap penyelesaian skripsi
15. Teman-teman Asisten Praktikum Basis Data 2016 yang telah memberikan pengalaman belajar mengajar sehingga ilmunya bisa diaplikasikan pada laporan skripsi
16. Semua pihak yang telah banyak membantu, berbagi ilmu dan pengalaman, serta memberikan dukungan selama proses pengerjaan skripsi yang tidak dapat penulis sebutkan satu per satu

Penulis mengakui bahwa dalam penyusunan skripsi ini masih terdapat banyak kekurangan, sehingga kritik yang membangun dan juga saran sangat penulis butuhkan. Akhir kata penulis berharap agar skripsi ini bisa membawa manfaat bagi semua pihak yang menggunakannya.

Wassalamualaikum warahmatullahi wabarakatuh.

Malang, 2 Agustus 2018

Penulis

tutiwardani@student.ub.ac.id

ABSTRAK

Tuti Wardani Hamid, Pengembangan Web Service Pada Iot Data Storage Dengan Pendekatan Semantik dan Penambahan Mekanisme Autentikasi

Dosen Pembimbing : Eko Sakti P., S.Kom., M.Kom. dan Reza Andria Siregar, S.T., M.Kom.

Internet of Things mempunyai sistem dan perangkat untuk memonitor objek, mengumpulkan, dan meneruskan data. Data yang beragam akan dikumpulkan menggunakan sensor kemudian diteruskan hingga tersimpan pada data storage. Untuk meneruskan data yang beragam meta dan ukuran tersebut, dibutuhkan sistem yang mampu menangani keberagaman data tersebut. Karena sebab itu, pada penelitian ini digunakan penyimpanan data yang mampu mengatasi permasalahan keberagaman data dan juga mampu berkomunikasi dengan *middleware* dan perangkat-perangkat lainnya. Penelitian ini berfokus untuk membangun web service pada sistem penyimpanan data sehingga dapat mempermudah proses penyimpanan data. Web service menggunakan konsep *Semantic Interoperability* yang mampu memperjelas makna dari data tersebut, yang diwujudkan dalam pengelompokan meta data dan ukuran data tertentu sebelum disimpan di data storage. Untuk mekanisme autentikasinya, digunakan JSON Web Token (JWT) untuk membatasi hak akses web service, sehingga untuk mengakses data dalam data storage diperlukan token sebagai langkah autentikasi. Kinerja web service akan diuji setelah diimplementasikannya konsep *Semantic Interoperability* dan mekanisme autentikasi. Adapun parameter ujinya yaitu, Penggunaan CPU, Penggunaan Memori, *Runtime*, *Throughput*, dan *Disk I/O*. Selain pengujian kinerja, juga dilakukan pengujian fungsionalitas dan skalabilitas dari web service itu sendiri. Hasil dari pengujian fungsionalitas yaitu fungsi web service pada umumnya bisa berjalan sesuai dengan fungsinya, serta web service mampu melakukan request data rata-rata 145 request, sedangkan untuk pengiriman data, rata-rata web service mampu mengambil 424 kali pengiriman data. Berdasarkan pengujian tersebut, sistem ini dapat menjadi solusi untuk melakukan penyimpanan data pada data storage.

Kata kunci: *Internet of Things*, *Web service*, *Semantic Interoperability*, JSON Web Token (JWT), *Data Storage*

ABSTRACT

Tuti Wardani Hamid, *The Development of A Web Service In Iot Data Storage With The Approach Of Semantics and Increase Authentication Mechanism*

Supervisors : Eko Sakti P., S.Kom., M.Kom. dan Reza Andria Siregar, S.T., M.Kom.

Internet of Things has systems and devices for monitoring object, collecting, and forwarding data. Heterogeneous data will collected using sensor and then forwarded until it saved on data storage. To forwarding that various type and size of data, needed systems that can be handle heterogeneous data. Therefore, this research will used data storage than can be solved the problem of heterogeneous data and can be communicated with middleware and other devices. This research has focused on building web service on data storage's system, so process of saving data will facilitate. This web service completed by Semantic Interoperability's concept that enable to obvious the meaning of data that represented in grouping of particular data types and size of data before saved in data storage. From authentication side, web service used JSON Web Token (JWT) for restricting privilege of web service, so that to access data in data storage needed token as authentication's step. The performance of web service will be tested after implementation Semantic Interoperability's concept and authentication's mechanism. As for testing parameter are Penggunaan CPU, Memory Usage, Runtime, Throughput, and Disk I/O. In addition to performance test, doing functionality test and scalability test from the web service. The result of functionality test is the web service can running as well according to web service's function, and web service enable to doing some request of data with average 145 requests, whereas for sending data, the average of web service enable to take 424 times for sending data. Based on that tests, this system can be solution for saving data on data storage.

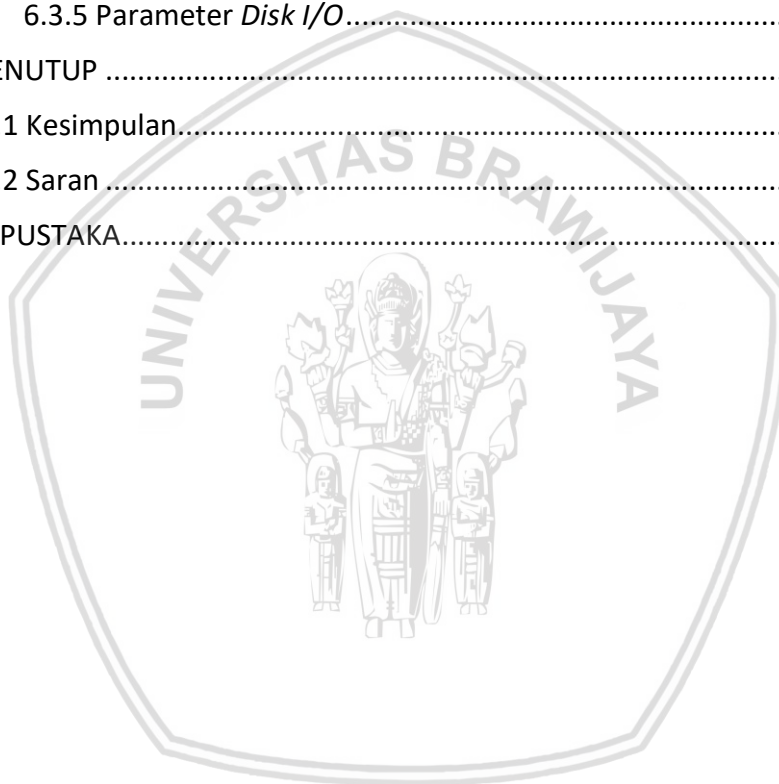
Keywords : *Internet of Things, Web service, Semantic Interoperability, JSON Web Token (JWT), Data Storage*

DAFTAR ISI

PENGESAHAN	ii
PERNYATAAN ORISINALITAS	iii
KATA PENGANTAR.....	iv
ABSTRAK.....	vi
ABSTRACT	vii
DAFTAR ISI	viii
DAFTAR TABEL.....	xi
DAFTAR GAMBAR.....	xii
BAB 1 PENDAHULUAN.....	1
1.1 Latar belakang.....	1
1.2 Rumusan masalah.....	3
1.3 Tujuan	3
1.4 Manfaat.....	3
1.5 Batasan masalah	3
1.6 Sistematika pembahasan.....	3
BAB 2 LANDASAN KEPUSTAKAAN	5
2.1 Kajian Pustaka	5
2.2 Dasar Teori.....	8
2.2.1 <i>Internet of Things (IoT)</i>	8
2.2.2 <i>Web Service</i>	8
2.2.3 <i>JSON Web Token (JWT)</i>	9
2.2.4 <i>Interoperability</i>	11
BAB 3 METODOLOGI	13
3.1 Studi Literatur	14
3.2 Analisis Kebutuhan	14
3.2.1 Kebutuhan Perangkat Keras.....	14
3.2.2 Kebutuhan Perangkat Lunak	14
3.2.3 Kebutuhan Data	15
3.3 Perancangan	15
3.4 Implementasi	16

3.5 Pengujian dan Pembahasan Hasil Pengujian	16
3.6 Kesimpulan dan Saran	16
BAB 4 ANALISIS KEBUTUHAN	17
4.1 Kebutuhan Pengguna.....	17
4.2 Lingkungan Operasi	17
4.3 Kebutuhan Data	17
4.4 Kebutuhan Sistem.....	17
4.4.1 Kebutuhan Fungsional.....	18
4.4.2 Kebutuhan Non Fungsional.....	18
BAB 5 PERANCANGAN DAN IMPLEMENTASI	20
5.1 Perancangan	20
5.1.1 Perancangan Lingkungan Sistem	20
5.1.2 Perancangan API RESTfull Web Service.....	21
5.1.3 Perancangan JSON Web Token (JWT).....	22
5.1.4 Perancangan Konsep <i>Semantic Interoperability</i>	24
5.1.5 Perancangan <i>IoT Application</i>	25
5.1.6 Perancangan Pengujian.....	27
5.2 Implementasi	30
5.2.1 Implementasi API RESTfull Web Service	30
5.2.2 Implementasi JSON Web Token.....	32
5.2.3 Implementasi Konsep <i>Semantic Interoperability</i>	33
5.2.4 Implementasi <i>IoT Application</i>	33
BAB 6 PENGUJIAN	35
6.1 Pengujian Fungsionalitas	35
6.1.1 Menerima Data <i>Text/JSON</i> Dari Internet <i>Gateway Device</i>	35
6.1.2 Menerima Data Gambar Dari Internet <i>Gateway Device</i>	36
6.1.3 Menyimpan Data Dari Internet <i>Gateway Device</i> ke Data <i>Storage</i>	37
6.1.4 Melakukan Autentikasi Dan Otorisasi Terhadap Subscriber Yang Akan Mengambil Data.....	38
6.1.5 Menyimpan Data dan Mengenali Meta Data Yang Akan disimpan ke Dalam Data <i>Storage</i>	42
6.1.6 Melakukan fungsi <i>POST</i> dan <i>GET</i> data	43

6.2 Pengujian Skalabilitas API Web Service	48
6.2.1 Pengujian <i>GET</i> Data	48
6.2.2 Pengujian <i>POST</i> Data	51
6.3 Pengujian Kinerja API Web Service	54
6.3.1 Parameter Penggunaan CPU	54
6.3.2 Parameter Penggunaan Memori	55
6.3.3 Parameter Runtime	56
6.3.4 Parameter <i>Throughput</i>	57
6.3.5 Parameter <i>Disk I/O</i>	58
BAB 7 PENUTUP	59
7.1 Kesimpulan	59
7.2 Saran	60
DAFTAR PUSTAKA	61



DAFTAR TABEL

Tabel 2.1 Kajian pustaka	6
Tabel 2.2 Kajian pustaka (lanjutan).....	7
Tabel 4.1 Kebutuhan fungsional	18
Tabel 4.2 Kebutuhan perangkat keras	18
Tabel 4.3 Kebutuhan perangkat lunak	19
Tabel 5.1 Skenario pengujian	28
Tabel 5.2 Skenario pengujian (lanjutan)	29
Tabel 5.3 Kode program pengimplementasian method <i>get data</i>	30
Tabel 5.4 Kode sumber pengimplementasian <i>post data</i>	31
Tabel 5.5 Kode sumber pengimplementasian token	32
Tabel 5.6 Kode sumber validasi token	32
Tabel 5.7 Kode sumber pengimplementasian konsep <i>semantic interoperability</i>	33
Tabel 6.1 Menerima data <i>text/JSON</i> dari IGD.....	35
Tabel 6.2 Menerima data gambar dari IGD	36
Tabel 6.3 Menyimpan data dari IGD ke data <i>storage</i>	37
Tabel 6.4 Uji autentikasi dan otorisasi	38
Tabel 6.5 Fungsi menyimpan dan mengenali data	42
Tabel 6.6 Melakukan fungsi <i>POST</i> dan <i>GET</i> data	43

DAFTAR GAMBAR

Gambar 2.1 Proses alur komunikasi JSON Web Token (JWT).....	10
Gambar 2.2 Isi dari Header Pada JWT.....	10
Gambar 2.3 Isi dari payload pada JWT.....	10
Gambar 2.4 Isi dari signature pada JWT	11
Gambar 3.1 Diagram alir metodologi peneltian	13
Gambar 5.1 Lingkungan sistem.....	20
Gambar 5.2 Sequence diagram sistem	21
Gambar 5.3 Diagram alir pembuatan akun.....	22
Gambar 5.4 Diagram alir login	23
Gambar 5.5 Diagram alir pembuatan token.....	24
Gambar 5.6 Diagram alir konsep <i>Semantic Interoperability</i>	25
Gambar 5.7 <i>Sequence</i> diagram <i>IoT Apps</i>	26
Gambar 5.8 Rancangan <i>interface IoT Apps</i>	26
Gambar 5.9 Rancangan halaman <i>register</i>	27
Gambar 5.10 Rancangan halaman <i>login</i>	27
Gambar 5.11 <i>Output</i> kode program <i>get data</i>	31
Gambar 5.12 <i>Output</i> kode program <i>post data</i>	32
Gambar 5.13 Tampilan halaman <i>register</i>	34
Gambar 5.14 Tampilan halaman <i>login</i>	34
Gambar 5.15 Tampilan halaman utama <i>IoT Application</i>	34
Gambar 6.1 Menerima data text/JSON dari IGD	36
Gambar 6.2 Menerima data gambar dari IGD	37
Gambar 6.3 Menyimpan data dari IGD ke data <i>storage</i>	38
Gambar 6.4 Proses login gagal.....	39
Gambar 6.5 Proses <i>login</i> berhasil	40
Gambar 6.6 Penggunaan token yang kadaluwarsa.....	40
Gambar 6.7 Penggunaan token yang tidak tepat	41
Gambar 6.8 Berhasil menggunakan token untuk melakukan fungsi <i>get data</i>	41
Gambar 6.9 Berhasil menggunakan token untuk melakukan fungsi <i>post data</i>	42
Gambar 6.10 Data berhasil disimpan di MongoDB.....	43

Gambar 6.11 Data berhasil disimpan di GridFS	43
Gambar 6.12 Hasil <i>get</i> data text	44
Gambar 6.13 Hasil <i>get</i> data gambar	45
Gambar 6.14 Salah satu data gambar yang diambil	45
Gambar 6.15 Berhasil <i>post</i> data text	46
Gambar 6.16 Data text telah tersimpan dalam data <i>storage</i>	46
Gambar 6.17 Data Gambar yang akan disimpan	47
Gambar 6.18 Berhasil <i>post</i> data gambar	47
Gambar 6.19 Data Gambar telah tersimpan dalam data <i>storage</i>	48
Gambar 6.20 Summary <i>report test</i> 50 request	49
Gambar 6.21 Summary <i>report test</i> 100 request	49
Gambar 6.22 Summary <i>report test</i> 150 request	50
Gambar 6.23 Summary <i>report test</i> 200 request	50
Gambar 6.24 Summary <i>report test</i> 250 request	51
Gambar 6.25 Summary <i>report test</i> 300 request	51
Gambar 6.26 Summary <i>report test</i> 100 kali	52
Gambar 6.27 Summary <i>report test</i> 300 kali	52
Gambar 6.28 Summary <i>report test</i> 500 kali	53
Gambar 6.29 Summary <i>report test</i> 700 kali	53
Gambar 6.30 Summary <i>report test</i> 1000 kali	54
Gambar 6.31 Rata-rata CPU usage ketika POST data	54
Gambar 6.32 Rata-rata CPU usage ketika GET data	55
Gambar 6.33 Rata-rata memory usage ketika POST data	55
Gambar 6.34 Rata-rata memory usage ketika GET data	56
Gambar 6.35 Runtime ketika POST data	56
Gambar 6.36 Runtime ketika GET data	57
Gambar 6.37 <i>Throughput</i> ketika POST data	57
Gambar 6.38 <i>Throughput</i> ketika GET data	58
Gambar 6.39 Disk I/O ketika POST data	58
Gambar 6.40 Disk I/O ketika GET data	58

BAB 1 PENDAHULUAN

1.1 Latar belakang

Internet of Things (IoT) telah menjadi teknologi yang cukup terkenal dengan kemampuannya saat diimplementasikan pada suatu aplikasi sejak beberapa tahun silam. Kelebihannya dalam memperoleh, mengintegrasikan, menyimpan, memproses, dan menggunakan data menjadi hal yang sangat penting dalam mencapai suatu tujuan akhir sejak sensor mulai menghasilkan data-data yang sangat besar (Cai, et al., 2017). IoT berfungsi untuk mengumpulkan data dan informasi dari beberapa *environment* yang selanjutnya akan diproses dengan cara yang berbeda-beda sesuai dengan kebutuhan. Umumnya, IoT mempunyai 3 elemen utama yaitu benda yang dilengkapi modul IoT, perangkat koneksi internet, dan *cloud data center* yang berfungsi sebagai penyimpanan aplikasi dan data (Arganata, 2017). Pada penelitian yang berjudul "*IoT-Based Big Data Storage Systems in Cloud Computing: Perspectives and Challenges*" terdapat 4 karakteristik data IoT yaitu data yang berasal dari sumber yang berbeda-beda, skala data besar yang dinamis, level data yang mempunyai semantik rendah, dan data yang tidak akurat. Data pada IoT bisa saja mempunyai volume yang semakin besar dan meta datanya akan semakin beragam, sehingga harus mampu mengelola semua data, baik data yang terstruktur maupun yang tidak terstruktur. Dengan adanya tantangan-tantangan tersebut, maka dibutuhkan *platform* yang mampu mendukung kemampuan penyimpanan dan pengolahan data yang terstruktur maupun tidak terstruktur secara efisien (Jiang & Xu, 2014).

Meta data yang dihasilkan juga sangat beragam, mulai dari *integer* hingga karakter, termasuk didalamnya data terstruktur maupun data yang tidak terstruktur. Komunikasi antara objek yang berbeda dalam lingkungan skala besar IoT yang dinamis menghasilkan volume yang besar secara *real time*, berkecepatan tinggi, serta data stream yang berkelanjutan. Data dari sensor juga mempunyai level yang mempunyai semantik rendah sebelum diproses. Padahal dibutuhkan semantik yang kompleks untuk mengabstraksikan data dari massa data level rendah. Beberapa penelitian juga menunjukkan bahwa sebagian besar sistem sensor hanya bisa mendapatkan 1/3 data yang akurat dikarenakan ketidakhandalan pembacaan data (Cai, et al., 2017).

Pada penelitian sebelumnya yang berjudul "Pengembangan Sistem Penyimpanan Data Berbasis MongoDB dan GridFS Untuk Menyimpan Data yang Beragam Dari *Node Sensor*" dibuatlah sistem penyimpanan data yang mampu menyimpan data besar secara efisien dan mendukung peningkatan kapasitas penyimpanan data (Arganata, 2017). Komunikasi antara *middleware* dengan data *storage* dibangun di atas sebuah perangkat yang bernama *Internet Gateway Device* (IGD) yang menjadi jembatan penghubung antara *middleware* dengan data *storage* sehingga informasi dari *middleware* bisa diberikan kepada data *storage* walaupun protokol komunikasinya berbeda. Selayaknya *Gateway* jaringan yang

berfungsi sebagai sistem yang menghubungkan dua jaringan dan bisa dikonfigurasi dalam aplikasi perangkat keras, lunak, ataupun keduanya (Kende, et al., 2015). Internet *Gateway Device* berperan sebagai *subscriber*, yang mana akan berlangganan data dari *middleware*. Dari Internet *Gateway Device*, data akan dikirim dan disimpan dalam data *storage*. Untuk mempermudah proses pengiriman dan pengambilan data maka dibangun sebuah API RESTful web *service*, sekaligus menjadi penghubung Internet *Gateway Device* dengan data *storage*. Selain disimpan pada data *storage*, data juga bisa dibaca dan ditampilkan pada sebuah web *application* yang selanjutnya disebut sebagai *IoT Application*. API RESTful Web *Service* yang dibangun mampu melakukan *post* data ke data *storage* dan melakukan *get* data yang telah tersimpan di data *storage*.

Namun dari penelitian sebelumnya, data dikelompokkan secara manual sebelum disimpan pada data *storage*. Data dikelompokkan sesuai dengan topik dari data tersebut, sehingga apabila terjadi penambahan topik, akan mengubah kode program yang ada pada sistem. Peneliti sebelumnya juga mengizinkan semua user untuk mengakses data *storage*, baik itu untuk menyimpan data ataupun untuk mengambil data. Perizinan hak akses tersebut dapat dilihat dari IoT *application* yang hanya menampilkan halaman untuk mengambil data pada data *storage* tanpa adanya autentikasi.

Berdasarkan permasalahan tersebut, pada penelitian ini akan berfokus pada pengembangan web *service* pada data *storage* yang merupakan pengembangan dari API RESTful web *service* yang dibuat oleh peneliti sebelumnya, yang mana web *service* akan dibuat pada mesin sendiri dan terpisah dengan *subscriber* dan IoT *application*. Web *service* juga ditambahkan fitur autentikasi menggunakan JWT (*JSON Web Token*) sehingga mampu melakukan fungsi autentikasi dan otorisasi untuk membatasi *subscriber* yang ingin mengakses data *storage*. Fungsi autentikasi dan otorisasi tersebut ditunjukkan dengan adanya sistem pendaftaran akun dan sistem *login* pada IoT *application* sebelum melakukan pengaksesan data *storage* untuk mendapatkan token. Token akan selalu digunakan pas saat melakukan penyimpanan dan pengambilan data. Diterapkan pula konsep *Semantic Interoperability*, yaitu suatu kemampuan suatu sistem untuk melakukan pertukaran data yang mempunyai berbagai macam makna. Konsep tersebut mampu mendukung proses pertukaran data yang beragam, sehingga tingkat akurasi suatu makna pada pertukaran informasi dapat lebih dipahami (European Research Cluster on the Internet of Things, 2015). Konsep tersebut dapat dilihat dari adanya fungsional mengenali meta dan ukuran data untuk menentukan jenis *database* yang digunakan, yaitu MongoDB dan GridFS. MongoDB akan digunakan untuk menyimpan meta data text dan berukuran kurang dari 800 kB, sedangkan GridFS digunakan untuk menyimpan meta data gambar dan berukuran lebih dari atau sama dengan 800 kB.

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah dibahas sebelumnya, maka rumusan masalah yang menjadi pokok penelitian yaitu :

1. Bagaimana menerapkan mekanisme autentikasi menggunakan JSON Web Token (JWT) pada API *Restful Web service* ?
2. Bagaimana menerapkan konsep *semantic interoperability* pada API RESTful Web service ?
3. Bagaimana menguji kinerja API RESTful Web service dengan adanya konsep *semantic interoperability* dan mekanisme autentikasi dan otorisasi ?

1.3 Tujuan

Tujuan dari penelitian ini :

1. Menerapkan mekanisme autentikasi menggunakan JSON Web Token (JWT) untuk membatasi pengguna dalam mengakses API RESTful Webservice.
2. Menyediakan web service yang berkonsep *Semantic Interoperability* yang mampu menerima data dari Internet Gateway Device (IGD) dan meneruskannya ke data storage.
3. Mengetahui bagaimana kinerja dari webservice setelah mempunyai konsep *Semantic Interoperability* dan telah memiliki mekanisme autentikasi.

1.4 Manfaat

Manfaat dari penelitian ini adalah untuk membangun web service yang mampu menjembatani Internet Gateway Device (IGD) dan data storage yang mempunyai fungsi *get* dan *post* data, autentikasi, dan mengenali metadata yang akan disimpan dalam data storage yang menggunakan konsep *Semantic Interoperability*.

1.5 Batasan masalah

Pada penelitian ini, fokus permasalahan yang akan dibahas yaitu :

1. Pengujian berfokus pada proses pertukaran data antara data yang ada di Internet Gateway Device (IGD) dan data storage.
2. Meta data yang digunakan saat pengujian yaitu JSON dan *image*.
3. Tempat penyimpanan data yang digunakan yaitu MongoDB dan GridFS.
4. Kapasitas data storage yaitu 16 GB.
5. Data yang digunakan adalah data gambar yang juga digunakan pada penelitian sebelumnya.

1.6 Sistematika pembahasan

Laporan penelitian ini mempunyai sistematika pembahasan dan penyusunan laporan yang dijelaskan sebagai berikut :

BAB 1 PENDAHULUAN

Bagian ini terdiri dari latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika pembahasan dari “Pengembangan Web Service Pada Iot Data Storage Dengan Pendekatan Semantik Dan Penambahan Mekanisme Autentikasi”.

BAB 2 LANDASAN KEPUSTAKAAN

Bagian ini membahas tentang dasar teori dari *Internet of Things*, *Web Service*, *RESTful Web service*, *JSON Web Token (JWT)*, dan *Semantic Interoperability* yang menjadi referensi dari penelitian ini.

BAB 3 METODOLOGI

Bagian ini berisi penjelasan terkait meta penelitian yang digunakan, metode yang digunakan, subjek penelitian dan lokasi penelitian yang dideskripsikan melalui penjelasan terkait alur penelitian yang didalamnya terdapat sub bab – sub bab dari laporan penelitian ini.

BAB 4 ANALISIS KEBUTUHAN

Bagian ini membahas kebutuhan apa saja yang dibutuhkan, baik secara fungsional, data dan non-fungsional, untuk mengimplementasikan web service sehingga proses implementasi dan pengujian dapat berjalan dengan baik.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Bagian ini membahas rancangan arsitektur, representasi data dalam model data dan basis data, serta penjelasan secara detail mengenai implementasi web service terhadap proses pertukaran data dari *Internet Gateway Device* sampai pada data storage sehingga dapat menunjukkan bahwa sistem bisa berjalan dengan baik.

BAB 6 PENGUJIAN

Bagian ini menunjukkan hasil dari pengujian dan membahas hasil pengujian secara keseluruhan apakah hasil tersebut sudah menjawab masalah-masalah yang ada pada rumusan masalah, sehingga dapat membuktikan bahwa sistem bisa berjalan dengan baik.

BAB 7 PENUTUP

Bagian ini berisi ringkasan dari pencapaian penelitian yang dilakukan yang berupa kesimpulan dari pembahasan rumusan masalah yang ada berdasarkan analisis dan pengujian serta berisi saran-saran dan pengembangan yang dapat dilakukan dari “Pengembangan Web Service Pada Iot Data Storage Dengan Pendekatan Semantik Dan Penambahan Mekanisme Autentikasi”.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Pada penelitian sebelumnya, telah dikembangkan suatu lingkungan IoT yang terdiri dari *node* sensor dan *middleware* yang memiliki mekanisme *publish subscribe*. *Middleware* mampu mengatasi permasalahan interoperabilitas dengan dukungan *gateway* yang menggunakan *multi-* protokol sehingga mampu menerima data dari berbagai jenis sensor (Anwari, 2017). Selanjutnya dikarenakan *middleware* hanya mempunyai kemampuan penyimpanan data yang bersifat sementara dan terbatas, sehingga dilakukan lagi pengembangan sistem IoT agar data yang telah didapatkan dapat dikelola sampai ke pusat data tanpa harus mengkhawatirkan volume data yang besar dan keberagaman data.

Persamaan dengan penelitian di atas yaitu memanfaatkan penggunaan *middleware* hasil penelitian di atas untuk mendapatkan data yang selanjutnya akan dikelola untuk disimpan di penyimpanan data, yang mana proses penyimpanan data tersebut akan dipermudah prosesnya dengan adanya *webservice*. Perbedaannya, penulis hanya menggunakan protokol komunikasi MQTT yang mengatur jalannya komunikasi antara sensor dan *middleware* dan protokol Websocket yang mengatur jalur untuk berkomunikasi dengan Internet Gateway Device (IGD).

Penelitian kedua yang digunakan sebagai pustaka berjudul “Pengembangan Sistem Penyimpanan Data Berbasis MongoDB dan GridFS Untuk Menyimpan Data yang Beragam Dari Node Sensor”. Penelitian ini mengembangkan sistem IoT mulai dari melakukan *subscribe* data dari *middleware* oleh Internet Gateway Device (IGD) hingga tersimpannya data dengan baik pada penyimpanan data. Penelitian ini berfokus pada pengembangan media penyimpanan data agar data yang beragam dan bervolume besar yang diterima dari *middleware* mampu tersimpan dengan baik (Arganata, 2017).

Persamaan dengan penelitian yang kedua yaitu penggunaan arsitektur yang sama mulai dari pengambilan data dari *middleware* hingga penyimpanan data yang sama yaitu MongoDB dan GridFS, yang mana proses pengambilan data dipermudah dengan adanya *webservice*. Perbedaannya, penulis hanya fokus untuk mengembangkan bagian *webservice* dari arsitektur sistem yang telah dibuat pada penelitian kedua. Bagian *webservice* diberi tambahan mekanisme otorisasi dan autentikasi, juga menggunakan konsep *semantic interoperability* sehingga data disimpan berdasarkan meta data dan ukuran data.

Tabel 2.1 Kajian pustaka

No.	Nama Penulis, Tahun, Judul	Persamaan	Perbedaan	
			Penelitian Terdahulu	Rencana Penelitian
1	Husnul Anwari, 2017, "Pengembangan IoT <i>Middleware</i> Berbasis Event-Based dengan Protokol Komunikasi CoAP, MQTT, dan Websocket"	Penggunaan <i>middleware</i> dan protokol MQTT dan websocket	Mengembangkan <i>middleware</i> sebagai <i>gateway</i> yang mampu menangani banyak protokol komunikasi pada pengembangan IoT	Memanfaatkan <i>middleware</i> yang telah dikembangkan sebagai <i>gateway</i> yang menggunakan protokol MQTT dan protokol websocket sebagai pengatur jalur komunikasi antara <i>middleware</i> dengan Internet Gateway Device
2	Gabreil Arganata, 2017, "Pengembangan Sistem Penyimpanan Data Berbasis MongoDB dan GridFS Untuk Menyimpan Data yang Beragam Dari Node Sensor"	Penggunaan aritektur sistem mulai dari pengumpulan data yang dilakukan oleh <i>middleware</i> hingga penyimpanan data yang dilakukan di MongoDB dan GridFS	Menggunakan <i>webservice</i> sederhana yang juga menggabungkan fungsi <i>subscribe</i> topik data di dalam program <i>webservice</i> tersebut	Membuat <i>webserice</i> yang dilengkapi dengan mekanisme otorisasi dan autentikasi sebelum mengakses media penyimpanan data, juga menerapkan konsep <i>semantic interoperability</i> sehingga data akan disimpan bukan berdasarkan topik, melainkan berdasarkan meta data dan ukuran data.

Tabel 2.2 Kajian pustaka (lanjutan)

No.	Nama Penulis, Tahun, Judul	Persamaan	Perbedaan	
			Penelitian Terdahulu	Rencana Penelitian
3	Andri Warda Pratama Putra, 2017, "Implementasi Autentikasi JSON Web Token (JWT) Sebagai Mekanisme Autentikasi Protokol Message Queuing Telemetry Transport (MQTT) Pada Perangkat NodeMCU"	Menggunakan autentikasi JSON Web Token (JWT) sebagai mekanisme autentikasi	Menggunakan JSON Web Token (JWT) sebagai mekanisme autentikasi untuk protokol MQTT yang terdapat pada perangkat Node MCU	Menggunakan JSON Web Token (JWT) sebagai mekanisme autentikasi untuk <i>Wet Service</i> yang diletakkan pada data storage
4	Yazid, 2018, "Analisis Komunikasi Antar IoT <i>Middleware</i> Dengan Node Sensor Kamera Berbasis Raspberry Pi Zero"	Membahas permasalahan <i>semantic interoperability</i> pada <i>Internet of Things</i>	Pengujian <i>semantic interoperability</i> pada <i>node</i> sensor berbasis kamera	Mengimplementasikan web <i>Service</i> dengan konsep <i>semantic interoperability</i> yang mampu mengatasi permasalahan pada konten dan konteks data yang dikirimkan
5	Kamarudin, Kusrini, Andi Sunyoto, 2018, "Uji Kinerja Sistem Web <i>Service</i> Pembayaran Mahasiswa Menggunakan Apache JMeter (Studi Kasus : Universitas AMIKOM Yogyakarta)"	Menguji performansi web <i>service</i> dengan parameter uji <i>throughput</i> menggunakan Apache JMeter	Melakukan pengujian dengan data yang sudah ditentukan dengan parameter uji <i>ramp-up periode</i> dan <i>response time</i>	Melakukan pengujian dengan data yang sudah ditentukan dengan parameter uji <i>Penggunaan CPU, Memory Usage, Disk I/O</i>

2.2 Dasar Teori

2.2.1 Internet of Things (IoT)

Internet of Things (IoT) merupakan paradigma baru yang berkembang sangat cepat dalam lingkup telekomunikasi nirkabel yang modern. Ide dasar dari konsep ini adalah persebaran berbagai jenis benda atau objek seperti *Radio-Frequency Identification* (RFID), sensor, aktuator, *mobile phone*, dan lain-lain yang melalui skema pengalamatan yang unik yang mampu berinteraksi satu sama lain dan bekerja sama dalam mencapai tujuan (Atzori, et al., 2010). Namun saat ini, Internet of Things bisa dikatakan bukan lagi sebagai teknologi terkini dikarenakan setiap harinya sangat banyak benda-benda yang telah tekoneksi dengan internet seperti kendaraan ataupun gedung. Diperkirakan pada tahun 2020 akan terdapat 20 miliar benda-benda yang akan terhubung satu sama lain dengan internet (Grgic, et al., 2016).

IoT merepresentasikan evolusi internet generasi selanjutnya dan memperluas kemampuannya untuk menggabungkan, menganalisis dan mendistribusi data yang akan diolah menjadi informasi dan ilmu pengetahuan. IoT bisa ditemukan hampir di seluruh area kehidupan manusia seperti kehidupan sosial, pariwisata, pendidikan, manajemen rantai suplai, militer, kesehatan, industri, dan lain-lain. Perangkat IoT mampu menghasilkan data sensor dalam jumlah besar untuk mencerminkan lingkungan fisik atau kondisi dari suatu benda atau manusia. Seperti yang sering dibawa oleh perangkat IoT yaitu sumber dan kapasitas penyimpanan data yang terbatas, sensor data butuh untuk ditransmisikan dan disimpan pada kerangka yang kaya akan *resource* seperti *cloud*. Di sisi lain, dalam proses analisis riwayat sensor data merupakan hal yang sangat terpenting dalam pembuatan keputusan pada berbagai macam aplikasi berbasis IoT (Li, et al., 2017).

2.2.2 Web Service

Web Service merupakan suatu layanan yang menyediakan suatu standar dan kemampuan beroperasi untuk komunikasi M2M (*machine to machine*) pada internet. Terdapat dua kelas utama pada Web Service yaitu *Representational State Transfer* (REST), yang mana sumber dayanya dimanipulasi menggunakan kumpulan operasi *stateless* yang seragam dan Web Service yang dinamis, yang menggunakan sekumpulan operasi yang acak seperti SOAP messages (Laine, 2011).

Dalam konteks *Internet of Things* (IoT), RESTful Web Service mempunyai beberapa keuntungan dibandingkan dengan Web Service yang dinamis seperti SOAP. Adapun beberapa kelebihan tersebut antara lain memiliki overhead yang sedikit, memiliki kompleksitas penguraian yang lebih sedikit, mudah dimodifikasi, dan mampu berintegrasi baik dengan HTTP. Sebagai nilai tambah, banyak aplikasi yang mendukung kinerja RESTful Web Service lebih baik pada sensor nirkabel dengan sumber daya yang terbatas sehingga menjadi lebih mudah dipahami dan lebih mudah diimplementasikan (Guinard, et al., 2011).

2.2.2.1 *RESTful Web Services*

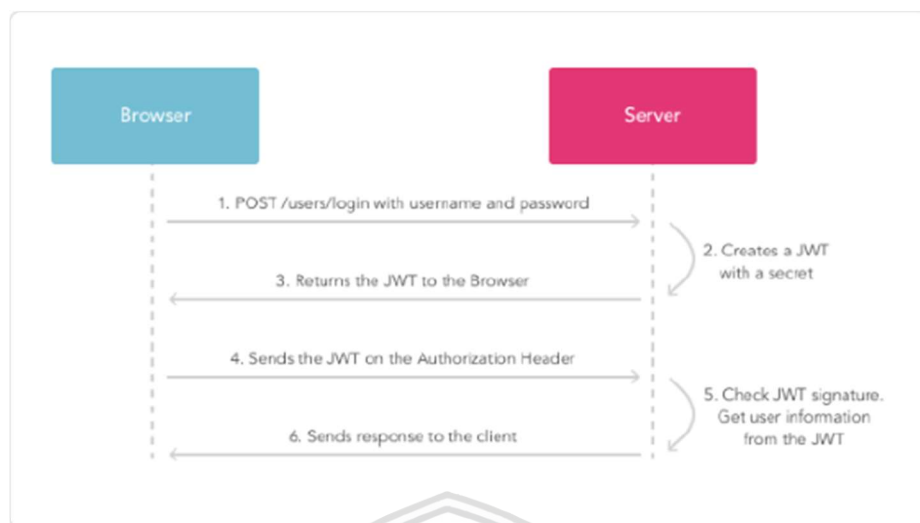
Representational State Transfer (REST) lebih dikenal secara meluas dalam penggunaan web sebagai alternatif yang lebih sederhana dibandingkan SOAP dan Web Services Description Language (WSDL). REST digambarkan sebagai sekumpulan prinsip arsitektur yang mampu merancang layanan web yang berfokus pada sumberdaya sistem, termasuk bagaimana sumberdaya dikelola dan ditransfer melalui HTTP dengan jangkauan client yang luas yang ditulis dalam bahasa yang berbeda. Faktanya, REST mempunyai dampak yang besar terhadap web yang hampir menggantikan SOAP dan WSDL yang berbasis perancangan antarmuka karena REST diketahui mempunyai model yang lebih sederhana untuk digunakan. Salah satu karakteristik dari *RESTful Web Service* yaitu secara eksplisit menggunakan metode HTTP yang didukung oleh protokol RFC 2616. Misalnya HTTP *GET* digambarkan sebagai metode penghasil data yang digunakan oleh aplikasi client untuk mendapatkan kembali sumberdaya, untuk mengambil data dari Web server, atau untuk mengeksekusi sebuah query dengan harapan bahwa web server akan melihat dan memberi respon sesuai dengan kumpulan sumberdaya lainnya (Rodriguez, 2015)

REST meminta developer untuk menggunakan metode HTTP dengan menggunakan protokol yang sesuai. Pada dasarnya, REST merancang prinsip yang menetapkan pemetaan satu sama lain antara operasi create, read, update, and delete (CRUD) dan metode HTTP. Berikut penjelasan terkait pemetaan tersebut :

1. Untuk membuat sumber pada server, gunakan kata kunci *POST*.
2. Untuk mendapat kembali sumber yang telah dibuat, gunakan kata kunci *GET*.
3. Untuk mengubah atau memperbaharui sumber, gunakan kata kunci *PUT*.
4. Untuk menghapus sumber, gunakan kata kunci *DELETE*.

2.2.3 JSON Web Token (JWT)

JSON Web Token (JWT) adalah klaim representasi format yang dibutuhkan untuk membatasi lingkungan seperti otorisasi header HTTP dan parameter query URI. JWT melakukan encode untuk dikirimkan sebagai objek JSON yang digunakan sebagai struktur payload dari Signature JSON Web atau sebagai sebuah struktur plaintext dari JSON Web Encryption (JWE), yang menjadikan *text* diberi tanda secara digital atau dilindungi dengan Message Authentication Code (MAC) (Haekal & Eliyani, 2016) . Dalam proses autentikasi, ketika pengguna selesai melakukan *login* dengan menginputkan perintah tertentu, JSON Web Token (JWT) akan dikembalikan dan akan disimpan di penyimpanan lokal. Gambar 2.1 menunjukkan proses alur komunikasi di dalam JSON Web Token (JWT).



Gambar 2.1 Proses alur komunikasi JSON Web Token (JWT)

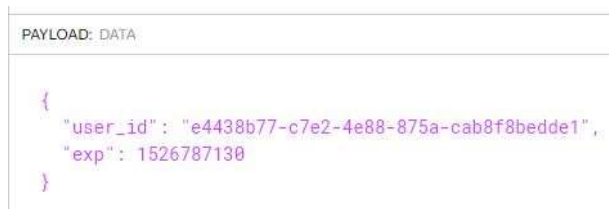
JWT terdiri dari tiga struktur yaitu :

- a. *Header* : berisikan informasi tentang jenis algoritme yang digunakan untuk membuat signature dengan memanfaatkan algoritme HMAC256. Adapun format header pada JSON ditunjukkan seperti Gambar 2.2. JSON tersebut akan di encode dengan algoritme base64 (Haekal & Eliyani, 2016).



Gambar 2.2 Isi dari Header Pada JWT

- b. *Payload* : berisikan informasi tentang entitas (pengguna) dan data tambahan berupa masa berlaku token seperti yang ditunjukkan pada Gambar 2.3 (Putra, 2017).



Gambar 2.3 Isi dari payload pada JWT

- c. *Signature* : struktur ini dibuat dari header dan payload yang digabungkan dan diencode menggunakan algoritme base64, kemudian dienkripsi menggunakan algoritme enkripsi HMAC-256. Contoh signature dari JWT dapat dilihat pada Gambar 2.4 .



Gambar 2.4 Isi dari signature pada JWT

Hasil dari tiga proses diatas akan digabungkan dan akan menghasilkan token. Pengguna mampu mengakses *webservice* dengan memasukkan token pada HTTP Header dalam metode otorisasi. Ketika client mengirimkan *request*, *webservice* akan melakukan verifikasi token terlebih dahulu. Setelah token telah diverifikasi server akan memeriksa masa berlaku dari token. Jika masa berlaku token telah berakhir, maka server memberikan informasi kepada client bahwa token sudah tidak dapat digunakan. Sebaliknya, jika masa berlaku token belum berakhir, server akan memeriksa pemberian akses yang tepat sesuai dengan isi dari *payload*. Jika pemberian akses benar, maka *webservice* akan merespon sesuai dengan kebutuhan *client* (Haekal & Eliyani, 2016).

2.2.4 Interoperability

Menurut Kamus Besar Bahasa Indonesia (2016), *Interoperability* atau Interoperabilitas adalah kemampuan berbagai jenis komputer, aplikasi, sistem operasi, dan jaringan untuk bertukar informasi dengan cara yang bermanfaat dan bermakna. Interoperabilitas merupakan kemampuan dua atau lebih sistem atau komponen untuk saling bertukar data dan menggunakan informasi. Pada definisi tersebut, terdapat beberapa tantangan yakni terkait bagaimana cara mendapatkan informasi, bagaimana cara bertukar data, dan bagaimana penggunaan informasi agar mudah dipahami dan kemampuan untuk memproses informasi tersebut (European Research Cluster on the Internet of Things, 2015).

2.2.4.1 Semantic Interoperability

Semantic interoperability biasanya terkait dengan format data. Pesan yang ditransfer menggunakan protokol komunikasi membutuhkan perumusan yang baik, meskipun jika itu hanya dalam bentuk bit. Namun, beberapa protokol membawa data atau konten dan bisa merepresentasikan sesuatu menggunakan *syntax* berlevel tinggi seperti HTML atau XML (European Research Cluster on the Internet of Things, 2015).

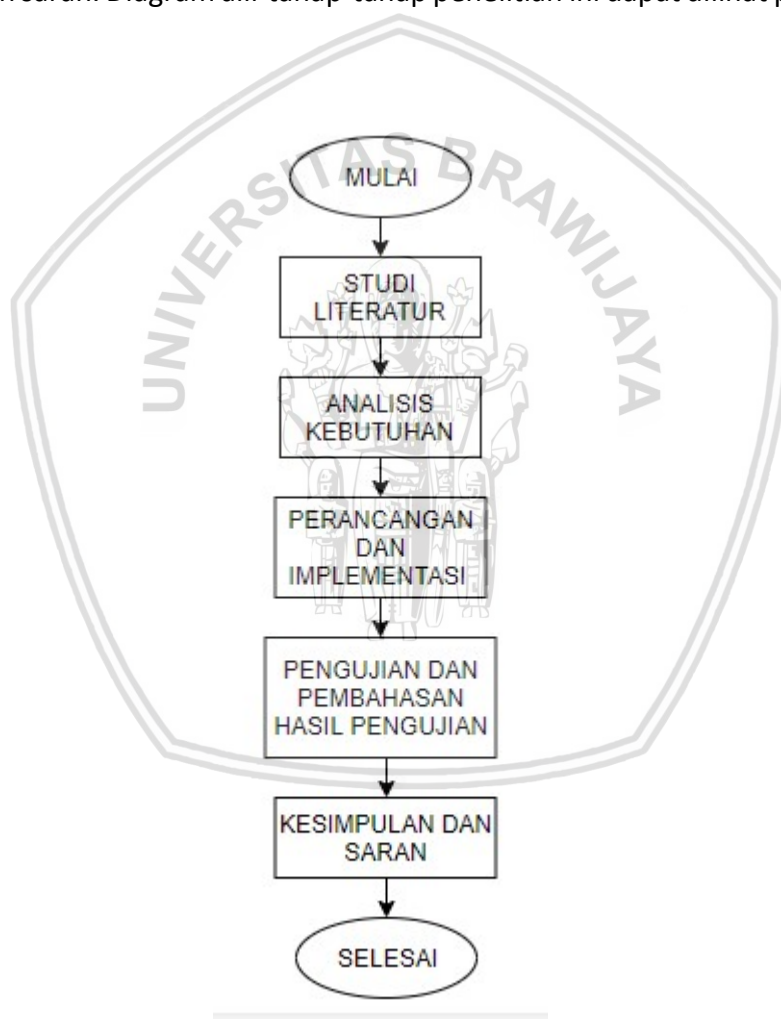
Semantic interoperability berarti setiap *stakeholder* yang berbeda bisa mengakses dan menginterpretasikan data secara jelas. Kata “*Things*” pada *Internet of Things* (IoT) membutuhkan pertukaran data antara satu sama lain dan dengan pengguna internet yang lain. Penyediaan deskripsi data yang jelas yang bisa diproses dan diinterpretasikan dengan mesin dan *software* merupakan pendukung informasi komunikasi secara otomatis dan interaksi dalam IoT. *Semantic* pada anotasi data bisa menyediakan.

Semantic interoperability mengacu pada konten dan konteks data yang dikirimkan. *Semantic interoperability* adalah proses pengiriman data mengacu pada konten dan konteks data yang dikirim. Pengujian *semantic* dilakukan dengan struktur data yang dikirim berbeda dari biasanya dan data yang dikirim berupa *text/JSON* dan Gambar.



BAB 3 METODOLOGI

Pada bab ini dijelaskan langkah-langkah serta metode yang akan digunakan dalam penelitian yang berjudul “Pengembangan Web Service Pada *lot Data Storage* Dengan Pendekatan Semantik Dan Penambahan Mekanisme Autentikasi”. Penelitian ini berawal dari dilakukannya studi literatur untuk memahami lebih jauh apa saja yang akan dilakukan pada penelitian ini. Selanjutnya dilakukan analisis terhadap kebutuhan apa saja yang dibutuhkan dalam melakukan perancangan dan implementasi. Lalu, setelah sistem telah diimplementasikan, akan dilakukan pengujian sistem. Hasil pengujian sistem tersebut akan dibahas pada tahap selanjutnya. Pada tahap akhir yang dilakukan yaitu pembuatan kesimpulan dan penulisan saran. Diagram alir tahap-tahap penelitian ini dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram alir metodologi penelitian

3.1 Studi Literatur

Sebagai penunjang penulisan skripsi, dibutuhkan studi literatur yang bertujuan untuk mencari dasar teori dan kajian pustaka. Dasar teori tersebut didapatkan dari beberapa buku, jurnal, dokumentasi project penelitian sebelumnya yang relevan dengan judul dan topik dari penelitian ini. Kajian pustaka yang digunakan yaitu beberapa penelitian terdahulu yang relevan dengan penelitian ini. Beberapa kajian pustaka yang digunakan yaitu penelitian dari Husnul Anwari yang berjudul “Pengembangan IoT *Middleware* Berbasis Event-Based dengan Protokol Komunikasi CoAP, MQTT, dan Websocket”, penelitian dari Gabreil Arganata yang berjudul “Pengembangan Sistem Penyimpanan Data Berbasis MongoDB dan GridFS Untuk Menyimpan Data yang Beragam Dari Node Sensor”, penelitian dari Yazid yang berjudul “Analisis Komunikasi Antar IoT *Middleware* Dengan Node Sensor Kamera Berbasis Raspberry Pi Zero”, dan penelitian dari Andri Warda Pratama Putra yang berjudul “Implementasi Autentikasi JSON Web Token (JWT) Sebagai Mekanisme Autentikasi Protokol Message Queuing Telemetry Transport (MQTT) Pada Perangkat NodeMCU”.

3.2 Analisis Kebutuhan

Pada langkah ini memasuki tahap yang membahas tentang kebutuhan-kebutuhan yang harus dipenuhi dalam suatu penelitian. Untuk pengerjaan penelitian ini terdapat dua kebutuhan yaitu kebutuhan perangkat lunak dan kebutuhan perangkat keras. Analisis kebutuhan sangat dibutuhkan dalam suatu penelitian dikarenakan perlunya diketahui terlebih dahulu hal-hal apa saja yang menjadi pendukung pengerjaan suatu penelitian, juga dapat menjadi acuan dalam perancangan sistem agar perancangan menjadi lebih terarah dan sistematis. Untuk menganalisis kebutuhan, dapat dimulai dengan melihat kepustakaan pada penelitian-penelitian sebelumnya yang sejenis yang dapat digunakan pada penelitian ini dan penelitian selanjutnya.

3.2.1 Kebutuhan Perangkat Keras

Perangkat keras yang diperlukan dalam penelitian ini yaitu perangkat keras yang juga digunakan pada penelitian sebelumnya. Terdapat sebuah Raspberry Pi yang berfungsi sebagai media yang digunakan untuk menjalankan *middleware*. Raspberry Pi didukung oleh beberapa perangkat pendukung seperti USB Adapter sebagai wireless adapter, mikrokontroler, modul DHT11/DHT22, juga modul kamera yang mana modul-modul tersebut berfungsi sebagai sensor yang bekerja menggunakan power adapter dengan output 12V/2.0A. Selanjutnya, terdapat pula sebuah server yang digunakan untuk menjalankan web *service* yang terhubung dengan data *storage*. Yang terakhir, terdapat sebuah laptop yang digunakan untuk mengakses virtual server dalam pengimplementasian web *service*.

3.2.2 Kebutuhan Perangkat Lunak

Selain perangkat keras, terdapat pula perangkat lunak yang dibutuhkan dalam pengerjaan penelitian ini, yaitu berupa putty sebagai aplikasi yang merupakan

aplikasi remote console /terminal yang digunakan meremote komputer dengan menggunakan port ssh untuk menjalankan *middleware* dan mengakses web *service*. Kebutuhan perangkat lunak lainnya yaitu bahasa pemrograman python yang dipilih sebagai bahasa yang tepat untuk membangun sistem yang berhubungan dengan jaringan. Terdapat pula library Flask dan library JWT sebagai pendukung bahasa pemrograman python dalam membangun web *service*. PyCharm sebagai aplikasi editor untuk *source code* yang dilengkapi dengan berbagai fitur *library* pendukung pembuatan web *service*.

3.2.3 Kebutuhan Data

Penelitian ini membutuhkan data yang berasal dari *middleware* yang akan diteruskan melalui *Internet Gateway Device* sebelum disimpan di *data storage*. Data yang akan disimpan ada dalam *data storage* merupakan data yang mempunyai meta data yang berbeda-beda dan juga ukuran data yang berbeda-beda. Data tersebut berupa *file* JSON dan *file* *pickle* yang akan diubah menjadi file JSON sebelum disimpan dalam *data storage*. Salah satu data yang dibutuhkan dalam penelitian ini merupakan data yang berasal dari sensor DHT11/22 yang datanya berupa data suhu dan kelembapan.

3.3 Perancangan

Pada penelitian ini, terdiri dari perancangan alur sistem dan perancangan API RESTful Web *service*. Perancangan alur sistem diperlukan untuk menggambarkan arsitektur sistem yang digunakan berdasarkan hasil analisis kebutuhan yang telah ditentukan. Dalam sistem yang digunakan, terdapat dua meta data yang dapat di *POST* oleh *webservice* dan akan disimpan di *data storage*. Data yang pertama berasal dari sensor DHT11/22 yang datanya berupa data suhu dan kelembapan. Data yang kedua bersumber dari sensor kamera yang menghasilkan data gambar. Kedua jenis data tersebut akan *unsubscribe* oleh *Internet Gateway Device* (IGD) lalu kemudian *dipost* ke web *service* untuk *diget* oleh *data storage* untuk disimpan. Tujuan dibuatnya suatu web *service* di antara *Internet Gateway Device* (IGD) dan *data storage* yaitu untuk memudahkan proses pengiriman data yang diambil dari IGD untuk disimpan di *data storage*.

Pada perancangan API RESTful Web *service*, terdapat aliran data model yaitu Client-Server. Client-Server terjadi dikarenakan adanya penggunaan protokol komunikasi *Hypertext Transfer Protocol* (HTTP) yang mendukung kinerja arsitektur komunikasi Representational State Transfer (REST). Aliran data client-server dapat dilihat ketika data hasil *subscribe* dari *middleware* yang dilakukan oleh IGD kemudian dilanjutkan dengan proses *POST* data pada web *service*, lalu data *storage* akan melakukan proses *GET* data untuk menyimpan data tersebut.

Pada perancangan pengujian, dijelaskan beberapa pengujian yang akan dilakukan, yaitu perancangan pengujian fungsional, perancangan pengujian skalabilitas, dan perancangan pengujian kinerja web *service*. Perancangan pengujian diperlukan agar pengujian lebih terarah dan tidak keluar batasan sehingga tujuan dari penelitian dapat tercapai.

3.4 Implementasi

Implementasi untuk penelitian kali ini diawali dengan menjalankan *middleware* yang telah dibuat oleh (Anwari, 2017), di mana ketika menjalankan *middleware* menghasilkan data yang berasal dari sensor suhu dan kelembaban dan sensor Gambar. Selanjutnya data akan diterima oleh Internet *Gateway Device* dapat dibaca oleh *web service*. *Web service* dapat mengetahui *subscriber* mana saja yang melakukan proses pengambilan data. Selanjutnya data tersebut akan disimpan ke dalam data *storage*. Proses implemetasi ini mencakup pembentukan mekanisme autentikasi dan otorisasi untuk mengaksesan *web service*, kemudian pengecekan data *storage* yang aktif, kemudian melakukan pengujian terkait fungsional yang telah dirancang sebelumnya juga dilakukan pengujian skalabilitas untuk melihat seberapa besar data yang bisa diambil dan seberapa banyak data yang bisa disimpan pada *data storage*.

3.5 Pengujian dan Pembahasan Hasil Pengujian

Pengujian dilakukan agar dapat diketahui apakah sistem dapat berjalan sesuai dengan kebutuhan dan perancangan sebelumnya. Pada pengujian kali ini terdapat tiga jenis pengujian yang akan dilakukan yaitu pengujian fungsional, pengujian kinerja *web service*, dan pengujian skalabilitas. Pengujian fungsional dilakukan berdasarkan kebutuhan fungsional sistem yang telah ditentukan. Jika semua kebutuhan fungsional sistem dapat berjalan dengan baik, maka akan dilanjutkan ke tahap pengujian skalabilitas. Jika pengujian kebutuhan fungsional belum berjalan dengan baik, maka akan dilakukan peninjauan kembali pada program atau perancangan, lalu kemudian menguji kembali pemenuhan kebutuhan fungsional sistem.

Untuk pengujian kinerja *web service*, sistem akan diuji kinerjanya ketika melakukan fungsi *post* dan *get* data, dengan parameter uji *Penggunaan CPU, Memory Usage, Disk I/O, Throughput, dan Runtime*. Untuk pengujian skalabilitas, sistem akan diberikan beban sistem sehingga akan terlihat kinerja sistem dalam menangani beban sistem yang berbeda-beda. Selanjutnya, data yang didapatkan dari hasil pengujian akan dianalisis agar dapat diketahui apakah sistem bekerja sesuai dengan yang diharapkan saat perancangan. Dalam hal ini, apakah seluruh kebutuhan fungsional sistem dapat berjalan sesuai dengan beban sistem yang diterapkan.

3.6 Kesimpulan dan Saran

Setiap penelitian pasti menghasilkan suatu kesimpulan berdasarkan rumusan masalah yang diselesaikan pada penelitian tersebut. Pada tahap ini dapat dipaparkan kesimpulan dari hasil pengimplementasian yang telah dilakukan beserta hasil kinerja sistem yang telah dikembangkan. Dapat dijelaskan juga apakah terdapat kesesuaian antara teori dan praktik dalam penyelesaian suatu masalah yang telah dirumuskan sebelumnya. Untuk pemberian saran ditujukan agar kekurangan-kekurangan yang terdapat dalam sistem yang telah dibuat dapat dibenahi dan juga dikembangkan lagi oleh peneliti selanjutnya.

BAB 4 ANALISIS KEBUTUHAN

4.1 Kebutuhan Pengguna

Dalam penelitian ini, terdapat beberapa kebutuhan yang diperlukan saat membangun sistem dan saat mengimplementasikan *web service* pada *data storage*. Adapun hal-hal yang dibutuhkan oleh pengguna yang akan menggunakan *web service* yaitu :

1. Pengguna dapat mengetahui data apa saja yang mampu *unsubscribe* dan disimpan di *data storage*.
2. Pengguna dapat melihat jenis *data storage* apa saja yang bisa dijangkau oleh *web service*.
3. Pengguna dapat mengakses sistem dimana saja dan kapan saja selama ada koneksi internet.
4. Pengguna dengan mudah melakukan *maintenance* / perubahan pada sistem jika diperlukan.

4.2 Lingkungan Operasi

Lingkungan operasi yang digunakan oleh *web Service* yaitu :

1. OS : Debian Linux 9 Stretch.
2. Prosesor : 900 MHz 32-bit quad-core ARM Cortex-A7.
3. RAM : 1GB.
4. *Storage* : 16GB atau lebih.

4.3 Kebutuhan Data

Salah satu data yang dibutuhkan dalam penelitian ini merupakan data yang berasal dari sensor DHT11/22 yang datanya berupa data suhu dan kelembapan. Meta data yang diterima dari sensor berupa *file JSON* yang kemudian akan diproses dan disimpan di *data storage* . Data lain yang dibutuhkan dalam penelitian ini yaitu data gambar yang diambil dari sensor kamera. Meta data gambar yang diterima oleh *middleware* berupa *pickle*, yang selanjutnya akan diubah menjadi *file JSON* ketika melewati *web service* sebelum data tersebut disimpan dalam *data storage* .

4.4 Kebutuhan Sistem

Kebutuhan sistem dideskripsikan dengan tujuan untuk mengetahui kebutuhan apa saja yang diperlukan saat melakukan perancangan sistem dan saat mengimplementasikan sistem pada *data storage* . Kebutuhan dalam penelitian ini dapat mempermudah proses perancangan dan implementasi, yang dibagi menjadi dua yaitu kebutuhan fungsional dan kebutuhan non fungsional.

4.4.1 Kebutuhan Fungsional

Kebutuhan fungsional merupakan kebutuhan-kebutuhan yang harus ada dalam sistem yang dibuat dan proses-proses yang dapat dilakukan oleh sistem. Dalam penelitian ini, web *service* harus mempunyai beberapa kebutuhan fungsional seperti yang terlihat pada Tabel 4.1.

Tabel 4.1 Kebutuhan fungsional

No.	Kebutuhan Fungsional
1	API Web <i>service</i> dapat menerima data <i>text/JSON</i> dari Internet <i>Gateway Device</i> .
2	API Web <i>service</i> dapat menerima data gambar dari Internet <i>Gateway Device</i> .
3	API Web <i>service</i> dapat menyimpan data dari Internet <i>Gateway Device</i> ke Data <i>Storage</i> .
4	API Web <i>service</i> dapat melakukan autentikasi dan otorisasi terhadap <i>subscriber</i> yang akan mengambil dan menyimpan data.
5	API Web <i>service</i> dapat menyimpan data dan mengenali meta data yang akan disimpan ke dalam data <i>storage</i> .
6	API Web <i>service</i> dapat melakukan fungsi <i>POST</i> dan <i>GET</i> data.

4.4.2 Kebutuhan Non Fungsional

Kebutuhan non-fungsional merupakan kebutuhan yang berfungsi sebagai batasan fungsi pada suatu sistem serta kebutuhan pendukung dalam perancangan dan implementasi sistem. Dalam hal ini terdapat kebutuhan perangkat keras dan kebutuhan perangkat lunak.

4.4.2.1 Kebutuhan Perangkat Keras

Kebutuhan perangkat keras yang diperlukan untuk mengimplementasikan web *service* seperti yang terlihat pada Tabel 4.2 .

Tabel 4.2 Kebutuhan perangkat keras

Perangkat	Keterangan
<i>Virtual Private Server</i> (VPS)	Perangkat ini digunakan untuk menjalankan web <i>service</i> .
Raspberry Pi	Perangkat ini digunakan untuk menjalankan <i>middleware</i> yang di dalamnya terdapat <i>node</i> sensor.
Toshiba Satellite S40Dt-A AMD A4	Perangkat ini digunakan untuk mengakses web <i>service</i> pada vps.

4.4.2.2 Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak untuk mengimplementasikan web *service* seperti yang terlihat pada Tabel 4.3 .

Tabel 4.3 Kebutuhan perangkat lunak

Perangkat	Keterangan
<i>Raspbian Jessie</i>	<i>Operating system</i> untuk Raspberry Pi
Putty	Aplikasi remote console / terminal yang digunakan untuk meremote komputer dengan menggunakan port ssh.
<i>Python</i>	Bahasa pemrograman yang digunakan untuk membangun sistem.
<i>Library Flask dan JWT</i>	Pendukung bahasa pemrograman python dalam membangun web <i>service</i> .
PyCharm	Aplikasi editor untuk kode sumber yang dilengkapi dengan banyak fitur <i>library</i> pendukung dalam pembuatan web <i>service</i> .
Apache JMeter	Aplikasi untuk menguji kapabilitas dan kinerja dari web <i>service</i> .
<i>Postman</i>	Aplikasi untuk menguji kemampuan web <i>Service</i> melakukan <i>post</i> dan <i>get</i> data.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

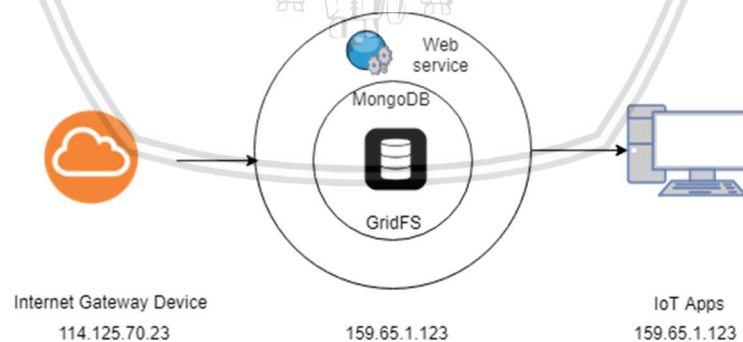
Pada bab sebelumnya, telah dilakukan analisis terhadap kebutuhan-kebutuhan untuk merancang dan mengimplementasi sistem. Keberhasilan suatu penelitian berawal dari perancangan sistem yang baik dan mengimplementasikannya sesuai dengan perancangan yang telah dibuat sebelumnya. Adapun perancangan yang akan dibuat meliputi, perancangan pada lingkungan sistem, perancangan API RESTful Webservice, perancangan data dan Perancangan Pengujian. Untuk implementasinya, akan dibuat implementasi sesuai dengan apa yang telah dirancang, yaitu implementasi API RESTful Webservice, implementasi JSON Web Token, dan implementasi *Semantic Interoperability*.

5.1 Perancangan

Perancangan dibuat untuk menjadi acuan dari pengimplementasian suatu penelitian sehingga proses implementasi dapat lebih terarah. Perancangan pada penelitian ini terdiri dari perancangan lingkungan sistem yang meliputi penjelasan arsitektur sistem yang digunakan beserta pengalamatan tiap sistem yang terhubung, juga alur kerja sistem yang mencerminkan bagaimana proses pengiriman data yang dilakukan oleh webservice.

5.1.1 Perancangan Lingkungan Sistem

Rancangan lingkungan sistem perlu dibuat agar sistem dapat berjalan sesuai dengan harapan. Rancangan lingkungan sistem ini menggambarkan bagaimana API RESTful Webservice menerima data dari Internet Gateway Device yang selanjutnya akan disimpan ke dalam data storage yaitu Mongo DB dan Grid FS. Rancangan tersebut dapat dilihat pada Gambar 5.1 .

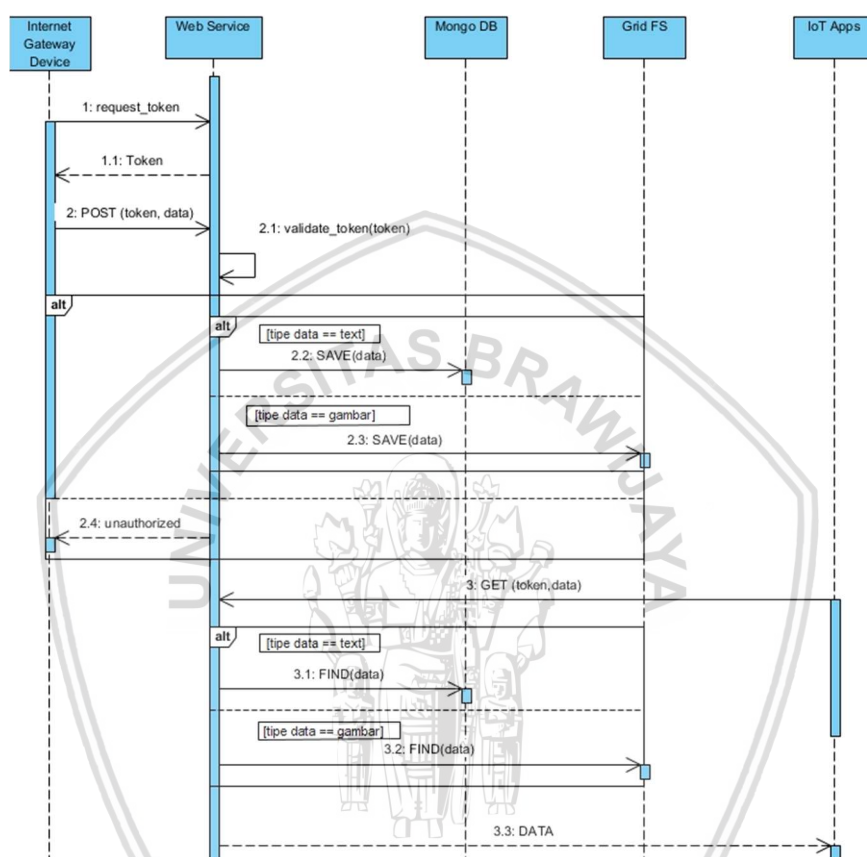


Gambar 5.1 Lingkungan sistem

Internet Gateway Device akan meneruskan data yang diterimanya dari *middleware* untuk disimpan di dalam data storage . Web service akan terlebih dahulu menerima data langsung dari Internet Gateway Device. Namun, pada penelitian ini, Web service melakukan pemfilteran data untuk disimpan di dalam data storage. Pemfilteran data dilakukan sebagai bentuk penerapan konsep *Semantic Interoperability*, yang mana data akan lebih terdeskripsikan dengan jelas karena mengacu pada konten dan konteks data yang dikirimkan.

Adapun data yang difilter berdasarkan meta data dan ukuran dari data tersebut. Untuk meta data *text* yang berukuran kurang dari 800 kB, akan disimpan dalam Mongo DB dan meta data gambar yang berukuran lebih dari atau sama dengan 800 kB akan disimpan dalam Grid FS. Selanjutnya, data yang telah tersimpan dalam data *storage* bisa dilihat melalui IoT Apps.

5.1.2 Perancangan API RESTfull Web Service



Gambar 5.2 Sequence diagram sistem

Rancangan API RESTful Webservice bisa dilihat pada Gambar 5.2 . Data yang diterima dari Internet Gateway Device akan diteruskan dan disimpan dalam data *storage* Mongo DB dan Grid FS. Seperti yang telah dijelaskan sebelumnya, untuk mempermudah pengambilan dan pengiriman data, maka dibuat sebuah API RESTful Web service. Namun, pada pembuatan API RESTful Web service kali ini ditambahkan mekanisme autentikasi dan otorisasi.

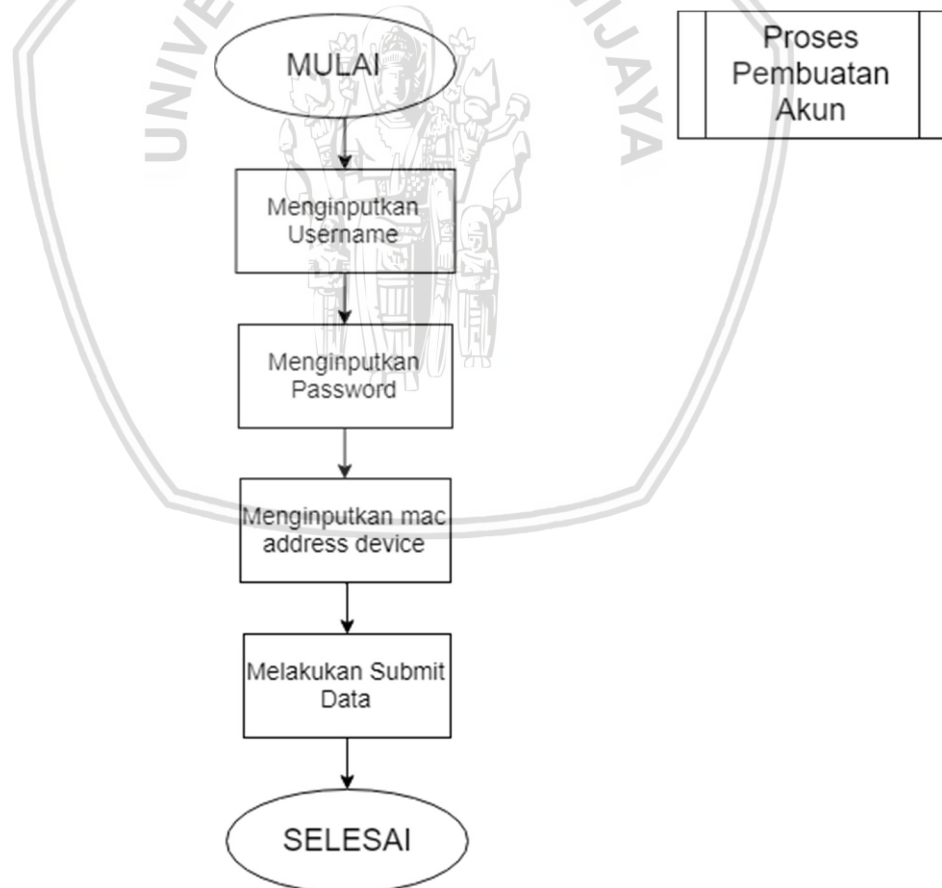
Proses autentikasi dan otorisasi yang dibangun dalam Webservice dimulai dengan dilakukannya request token oleh Internet Gateway Device ketika ingin meneruskan data untuk disimpan pada data *storage* . Lalu setelah mendapatkan token dari Web service, data akan diteruskan untuk disimpan dalam data *storage*. Selanjutnya, data akan disimpan dalam data *storage*.

Proses penyimpanan data dalam data *storage* menggunakan konsep *Semantic Interoperability* yang ditunjukkan dengan penyimpanan data berdasarkan meta

data. Pada penelitian ini, data yang bermeta *text* akan disimpan dalam MongoDB sedangkan data yang bermeta *image* disimpan dalam GridFS. Selanjutnya, data yang telah tersimpan dalam data *storage* dapat dilihat pada *IoT Apps*. *IoT Apps* meminta akses data *storage* pas web *Service*, kemudian web *Service* akan mencari data sesuai dengan permintaan *IoT Apps*. Setelah data ditemukan, data akan dikirim ke *IoT Apps* untuk ditampilkan.

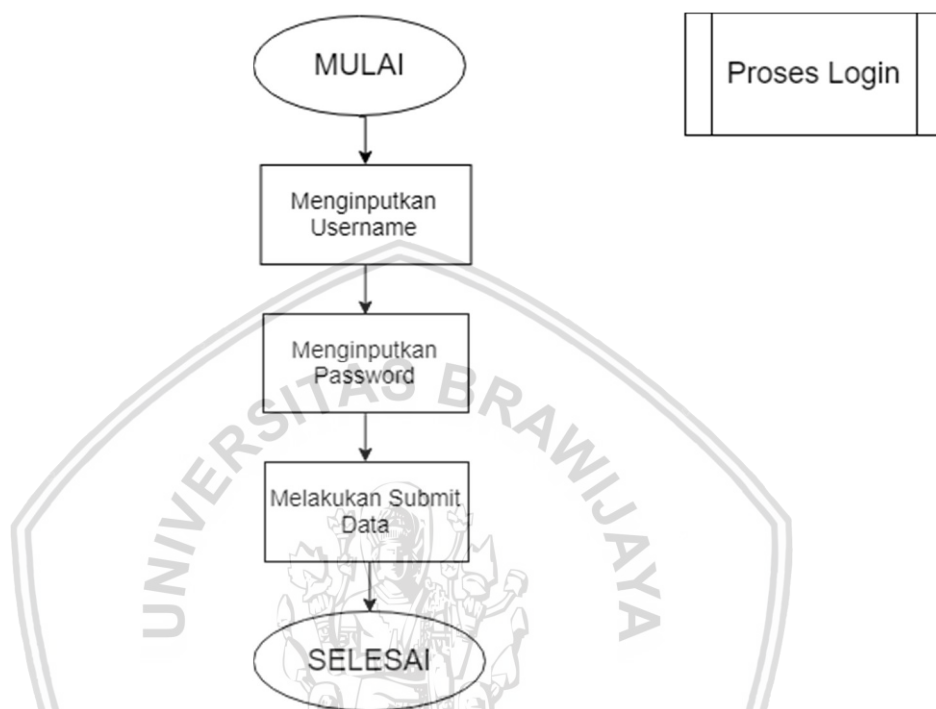
5.1.3 Perancangan JSON Web Token (JWT)

Untuk mendapatkan token, setiap pengguna yang ingin mengakses data dari data *storage* harus mempunyai akun terlebih dahulu. Akun dibuat dengan menginputkan *username*, *password*, dan *mac address* dari *device* yang digunakan. *Username* yang dimasukkan bisa berupa nama pengguna atau nama yang mudah diingat dikarenakan *username* yang dimasukkan akan digunakan juga untuk *login*. *Password* yang dimasukkan sebaiknya kombinasi huruf dan angka dan tentu saja mudah untuk diingat. *Password* tersebut akan digunakan ketika melakukan *login*. *Mac address* yang dimasukkan merupakan *mac address* dari *Internet Gateway Device* yang digunakan. Diagram alir untuk proses dapat dilihat pada Gambar 5.3.



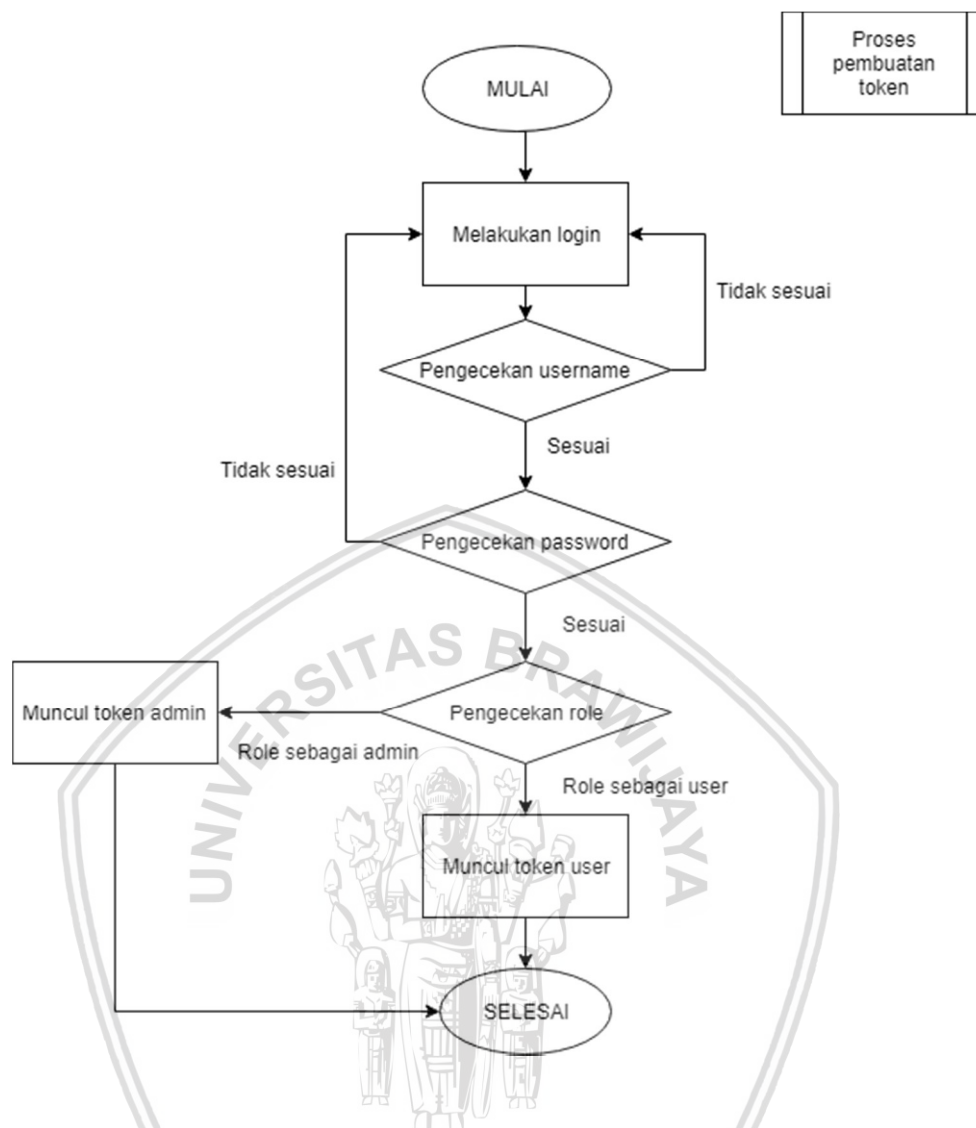
Gambar 5.3 Diagram alir pembuatan akun

Setelah memiliki akun, pengguna harus melakukan *login* terlebih dahulu agar bisa mendapatkan token untuk memiliki hak akses baik untuk melakukan *post* data ataupun *get* data. Diagram alir untuk *login* akun dapat dilihat pada Gambar 5.4 . *Login* dilakukan dengan cara menginputkan *username* dan *password* yang telah didaftarkan pada proses pembuatan akun sebelumnya.



Gambar 5.4 Diagram alir login

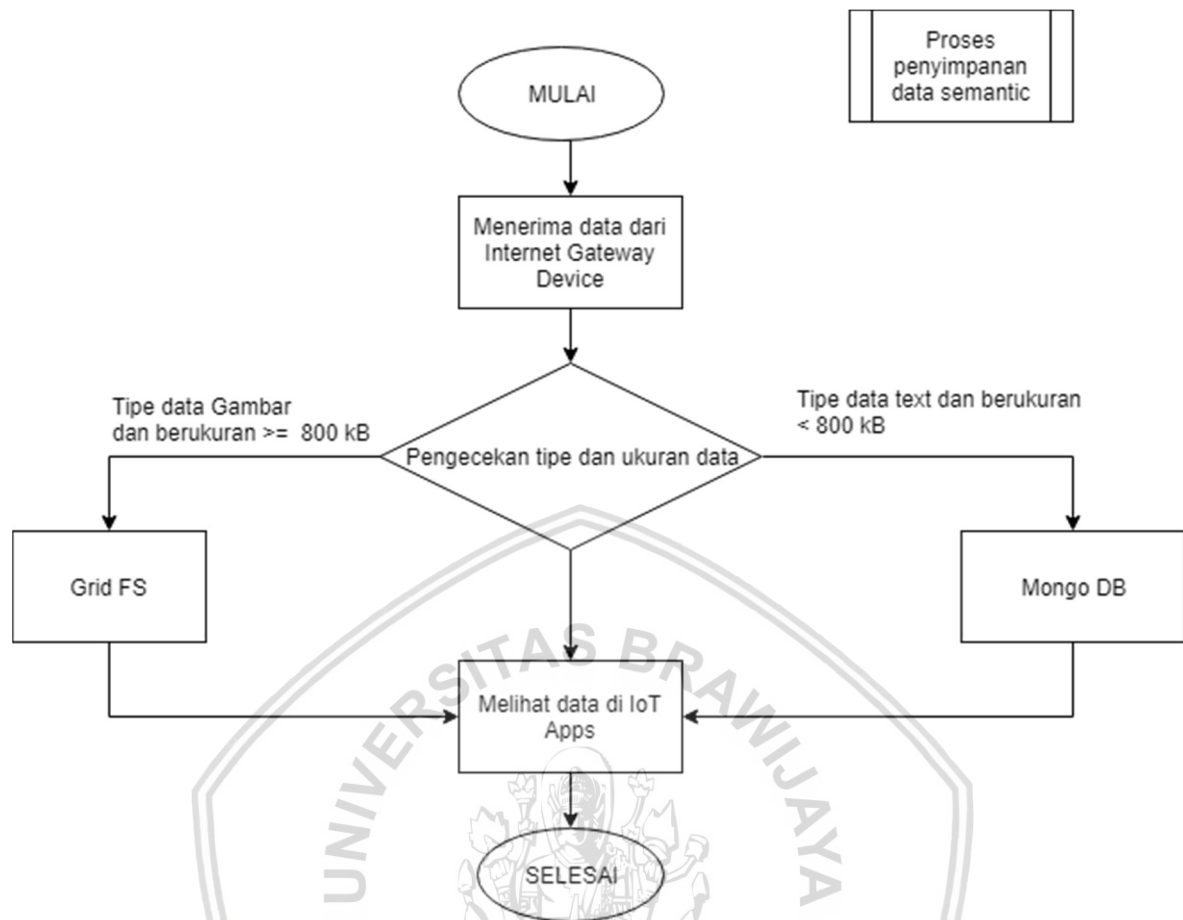
Setelah melakukan *login*, pengguna akan mendapatkan token untuk mengakses data *storage* . Diawali dengan pengecekan *username* dan *password*, jika *username* dan *password* yang dimasukkan ketika *login* sesuai dengan *username* dan *password* yang dimasukkan ketika membuat akun, maka selanjutnya akan dilakukan pengecekan *role*. Jika *role* sebagai *user* maka akan diberikan token untuk *user*. Jika yang melakukan *login* merupakan *admin*, maka akan diberikan token untuk *admin*. Namun apabila pada saat pengecekan *username* dan *password* ditemukan perbedaan pada *username* dan *password* yang dimasukkan ketika *login* dengan *username* dan *password* yang dimasukkan ketika membuat akun, maka pengguna harus melakukan *login* kembali dengan memasukkan *username* dan *password* sesuai dengan *username* dan *password* yang dimasukkan ketika membuat akun. Diagram alir untuk proses pembuatan token dapat dilihat pada Gambar 5.5 .



Gambar 5.5 Diagram alir pembuatan token

5.1.4 Perancangan Konsep *Semantic Interoperability*

Perancangan untuk konsep *semantic interoperability* diperlukan agar konsep ini dapat diimplementasikan dengan baik. Sebelumnya, data diterima dalam berbagai macam meta data dan ukuran data. Konsep *semantic interoperability* ditunjukkan dengan adanya pengelompokan data sesuai dengan meta data dan ukuran data. Konsep *semantic interoperability* dapat diimplementasikan sesuai dengan diagram alir pada Gambar 5.6 . Ketika data telah diterima dari *Internet Gateway Device* dalam berbagai macam meta data dan ukuran data, *web service* terlebih dahulu melakukan pengecekan meta data dan ukuran data sebelum data tersebut disimpan dalam data *storage*. Untuk meta data gambar dan data yang berukuran ≥ 800 kB akan disimpan pada GridFS. Sedangkan meta data *text* dan data yang berukuran < 800 kB akan disimpan pada MongoDB. Data yang tersimpan pada data *storage* bisa dilihat melalui *IoT Apps*.

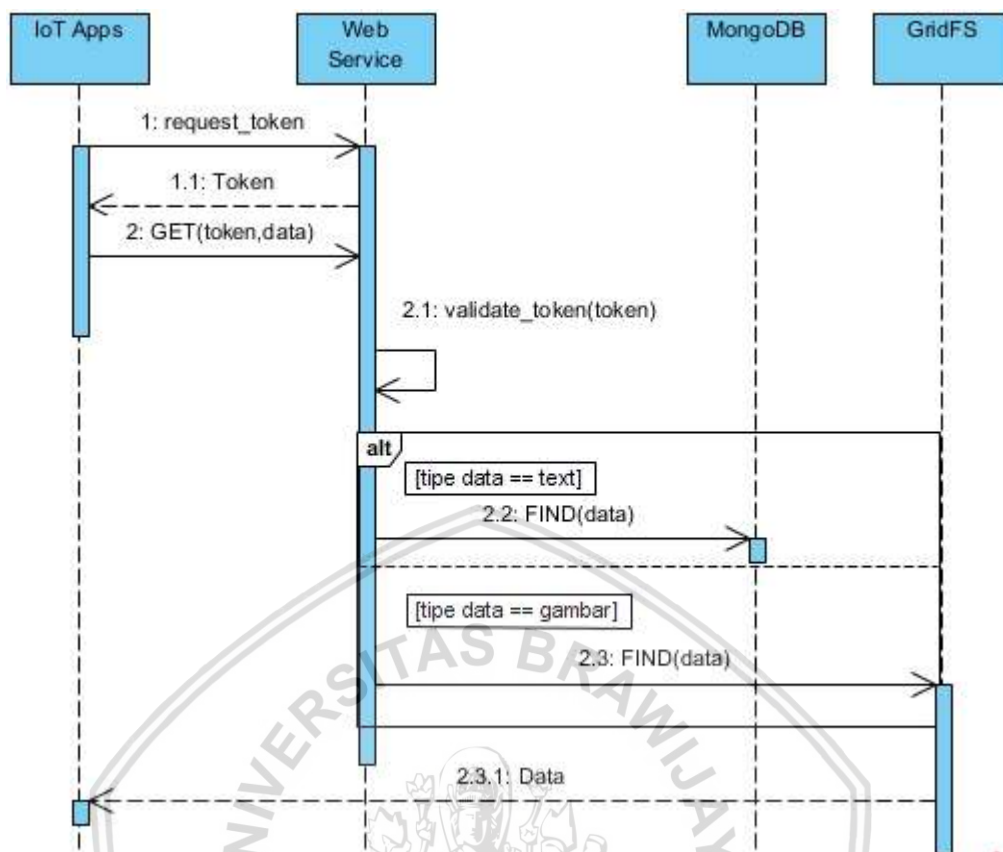


Gambar 5.6 Diagram alir konsep *Semantic Interoperability*

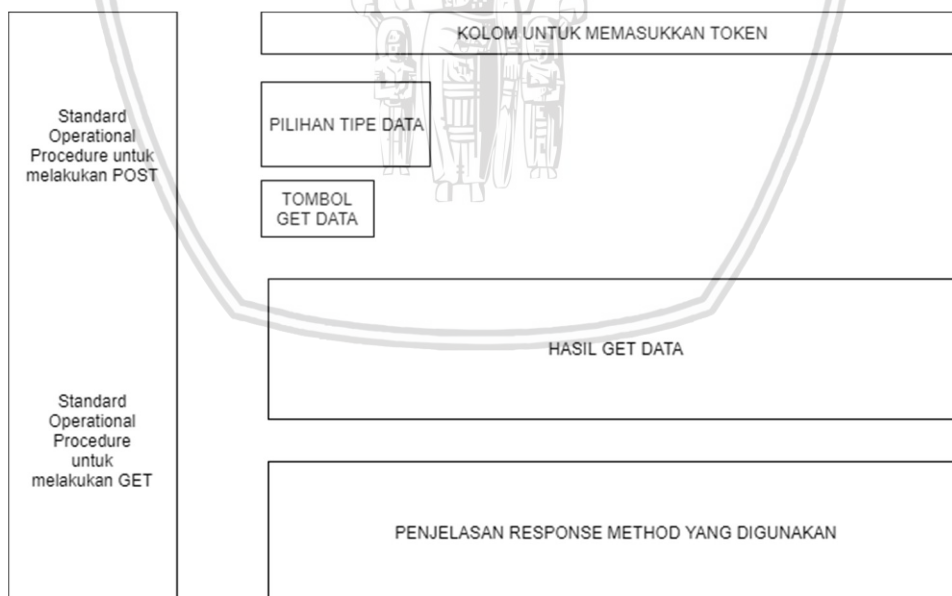
5.1.5 Perancangan *IoT Application*

IoT Apps dibutuhkan sebagai media untuk menampilkan data yang telah tersimpan pada data *storage*. Rancangan *IoT Apps* dapat dilihat pada Gambar 5.7. Untuk menampilkan data, terlebih dahulu *IoT Apps* melakukan permintaan token dan data pada *web service*. Setelah memberikan token pada *IoT Apps*, token akan divalidasi terlebih dahulu sebelum digunakan. Setelah memasukkan token, *web service* akan mencari data sesuai dengan permintaan *IoT Apps*. Setelah menemukan data sesuai dengan permintaan *IoT Apps*, data selanjutnya dikirimkan ke *IoT Apps* untuk ditampilkan.

Rancangan tampilan *IoT Apps* dapat dilihat pada Gambar 5.8 yang menampilkan tahap untuk melakukan *post* dan *get* data, kolom untuk memasukkan token, pilihan meta data, dan tombol *get* untuk menampilkan data. Di bagian bawah berisikan penjelasan tentang respons yang akan didapatkan ketika melakukan *get* dan *post* data.



Gambar 5.7 Sequence diagram IoT Apps



Gambar 5.8 Rancangan interface IoT Apps

IoT Apps dilengkapi dengan autentikasi terlebih dahulu sebelum masuk ke halaman untuk mengakses data. Diawali dengan halaman *register* untuk pembuatan akun terlebih dahulu yang ditunjukkan pada Gambar 5.9 yang menampilkan kolom untuk memasukkan *username*, *password*, dan *mac address*

dari *device* yang digunakan. Selanjutnya *login* dapat dilakukan melalui halaman yang ditunjukkan pada Gambar 5.10 yang menampilkan kolom untuk memasukkan *username* dan *password* untuk *login*.

REGISTER

KOLOM MEMASUKKAN USERNAME
KOLOM MEMASUKKAN PASSWORD
KOLOM MEMASUKKAN MAC ADDRESS
TOMBOL DAFAR

Gambar 5.9 Rancangan halaman *register*

LOGIN

KOLOM MEMASUKKAN USERNAME
KOLOM MEMASUKKAN PASSWORD
TOMBOL LOGIN

Gambar 5.10 Rancangan halaman *login*

5.1.6 Perancangan Pengujian

Perancangan pengujian diperlukan agar dapat diketahui batas minimal pengujian dan melakukan penelitian berdasarkan skenario pengujian yang telah dibuat.

5.1.6.1 Pengujian Fungsional

Pengujian fungsional bertujuan agar dapat diketahui apakah sistem yang dibuat telah memenuhi kebutuhan fungsional atau tidak. Jika seluruh kebutuhan fungsional telah terpenuhi maka dilanjutkan ke pengujian berikutnya. Namun jika kebutuhan fungsional belum memenuhi kebutuhan fungsional secara keseluruhan, maka akan dilakukan pengecekan kembali rancangan sistem dan juga kode sistem yang digunakan, lalu diadakan pengujian sekali lagi. Hal itu dilakukan terus menerus sampai semua kebutuhan fungsional tercapai. Untuk skenario pengujian, bisa dilihat pada Tabel 5.1 . Pada Tabel 5.1 terdapat kolom kode yang berawalan “WS” yang menandakan bahwa pengujian yang dilakukan adalah pengujian fungsionalitas dari web *service* diikuti dengan angka yang menunjukkan urutan pengujian. Kolom fungsi berisikan fungsi-fungsi yang harus dijalankan oleh web *service*. Fungsi tersebut dijalankan sesuai dengan skenario pengujian yang telah dituliskan sesuai dengan fungsinya masing-masing.

Tabel 5.1 Skenario pengujian

Kode	Fungsi	Skenario
WS_001	API Web service dapat menerima data <i>text/JSON</i> dari Internet Gateway Device	<ol style="list-style-type: none"> 1. <i>Middleware</i> dijalankan 2. Pengguna menjalankan IGD 3. Pengguna melihat data <i>text/JSON</i> yang masuk melalui Putty
WS_002	API Web service dapat menerima data gambar dari Internet Gateway Device	<ol style="list-style-type: none"> 1. <i>Middleware</i> dijalankan 2. Pengguna menjalankan IGD 3. Pengguna melihat data gambar yang masuk melalui Putty
WS_003	API Web service dapat menyimpan data dari Internet Gateway Device ke Data Storage	<ol style="list-style-type: none"> 1. <i>Middleware</i> dijalankan 2. Pengguna menjalankan IGD 3. Pengguna melihat data yang telah masuk ke data storage melalui RoboMongo
WS_004	API Web service dapat melakukan autentikasi dan otorisasi terhadap <i>subscriber</i> yang akan mengambil dan menyimpan data	<ol style="list-style-type: none"> 1. <i>Middleware</i> dijalankan 2. Pengguna menjalankan IGD 3. Pengguna melakukan <i>login</i> menggunakan <i>username</i>, <i>password</i>, dan <i>mac address</i> yang telah didaftarkan untuk mendapatkan token 4. Pengguna mengambil dan menyimpan data menggunakan token yang telah didapatkan
WS_005	API Web service dapat menyimpan data dan mengenali meta data yang akan disimpan ke dalam data storage	<ol style="list-style-type: none"> 1. <i>Middleware</i> dijalankan 2. Pengguna menjalankan IGD 3. Pengguna menggunakan token yang sudah didapatkan setelah <i>login</i> untuk melakukan penyimpanan data 4. Data akan dipisahkan tempat penyimpanan datanya sesuai dengan meta data dan ukuran datanya

Tabel 5.2 Skenario pengujian (lanjutan)

Kode	Fungsi	Skenario
WS_006	API Web service dapat melakukan fungsi <i>POST</i> dan <i>GET</i> data	<ol style="list-style-type: none"> 1. <i>Middleware</i> dijalankan 2. Pengguna menjalankan IGD 3. Pengguna melakukan <i>login</i> menggunakan <i>username</i>, <i>password</i>, dan <i>mac address</i> yang telah didaftarkan untuk mendapatkan token 4. Pengguna mengambil dan menyimpan data (melakukan fungsi <i>get</i> dan <i>post</i>) menggunakan token yang telah didapatkan

5.1.6.2 Pengujian Skalabilitas

Pengujian skalabilitas bertujuan untuk mengetahui batas kemampuan suatu sistem dalam menangani beban sistem sehingga sistem tersebut akan digunakan sesuai dengan kemampuannya dalam menangani beban sistem (Arganata, 2017). Pengujian skalabilitas dilakukan dengan cara melakukan pengambilan data dengan jumlah *request* data yaitu 50 *request*, 100 *request*, 150 *request*, 200 *request*, 250 *request*, dan 300 *request*. Selain pengambilan data, dilakukan pula uji coba penyimpanan data yang sama sebanyak 100 kali, 300 kali, 500 kali, 700 kali, dan 1000 kali. Ketika proses pengambilan data dan penyimpanan data berlangsung, dapat dilihat berapa banyak beban sistem yang dapat ditangani oleh web service.

5.1.6.3 Pengujian Kinerja API RESTful Webservice

Pengujian kinerja dilakukan untuk mengetahui bagaimana kinerja API RESTful Web service ketika digunakan (Kamarudin, et al., 2018). Adapun parameter dalam uji kinerja penelitian ini yaitu :

1. Penggunaan CPU : persentase penggunaan prosesor ketika proses pengiriman dan pengambilan data berlangsung.
2. Penggunaan Memori: persentase penggunaan memori ketika proses pengiriman dan pengambilan data berlangsung.
3. *Throughput* : jumlah operasi yang dieksekusi oleh web service setiap detiknya.
4. *Runtime* : waktu yang digunakan untuk menyimpan atau mengambil data.
5. *Disk I/O* : kinerja disk ketika proses pengambilan dan penyimpanan data dilakukan.

Pengujian kinerja dengan parameter tersebut akan dilakukan bersamaan dengan uji skalabilitas dimana jumlah *request* data yang digunakan dan jumlah data yang akan disimpan bernilai sama dengan uji skalabilitas. Ketika *web service* dijalankan, parameter yang telah ditentukan akan menunjukkan nilai dalam bentuk grafik. Khusus untuk penggunaan CPU, memori, dan *disk I/O* akan diambil nilai rata-ratanya untuk dituliskan pada hasil pengujian.

5.2 Implementasi

Implementasi merupakan hal yang paling penting dari sebuah penelitian. Implementasi dilakukan untuk memenuhi tujuan penelitian yang berdasarkan pada perancangan yang telah dilakukan sebelumnya. Adapun implementasi yang akan dilakukan yaitu implementasi API RESTful *Web service*, implementasi JSON Web Token (JWT), dan implementasi konsep *Semantic Interoperability*. Hasil pengimplementasian tersebut akan diujikan pada bab berikutnya yaitu bab pengujian.

5.2.1 Implementasi API RESTfull Web Service

Pembuatan *webservice* ditulis menggunakan bahasa pemrograman python dengan menggunakan *library Flask*.

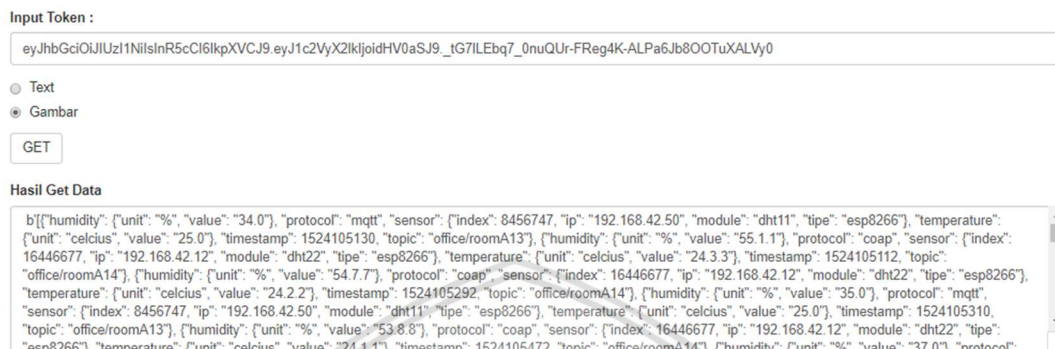
5.2.1.1 Methods GET

Pengimplementasian *webservice* untuk method *GET* bisa dilihat pada Tabel 5.2. Untuk mengambil data dari data *storage*, perlu dilakukan inialisasi terlebih dahulu database yang digunakan. Diawali dengan proses koneksi, dan beberapa kondisi yang akan diterapkan, proses *get* data bisa dilakukan. Untuk menjalankannya, bisa dengan memasukkan alamat <https://iot.bermainmainditaman.com:5001/api/getdata> . Namun harus memenuhi ketentuan hak akses yaitu memasukkan token terlebih dahulu.

Tabel 5.3 Kode program pengimplementasian method *get* data

Method : <i>getdata</i>	
1	@app.route('/api/getdata',methods=['GET'])
2	@token_required()
3	def getdata(self):
4	db_ = client.test
5	data = []
6	if request.args.get('topic') :
7	if request.args.get('topic')== "Gambar":
8	fs = gridfs.GridFS(db_)
9	for data_ in fs.find():
10	
11	data.append("https://iot.bermainmainditaman.com:5001/api/getGambar
12	/"+str(data_._id))
13	else:
14	for data_ in db_.testH__130.find({"topic":
15	request.args.get('topic')},{ "_id":0}):
16	data.append(data_)
17	else:
18	for data_ in db_.testH__130.find({}, {"_id": 0}):
19	data.append(data_)
20	return json.dumps(data, sort keys=True)

Hasil dari keluaran kode program implementasi *get* data dapat dilihat pada Gambar 5.11 yang menampilkan data yang berhasil diambil yang sebelumnya harus melewati ketentuan hak akses berupa penggunaan token.



Gambar 5.11 Output kode program *get* data

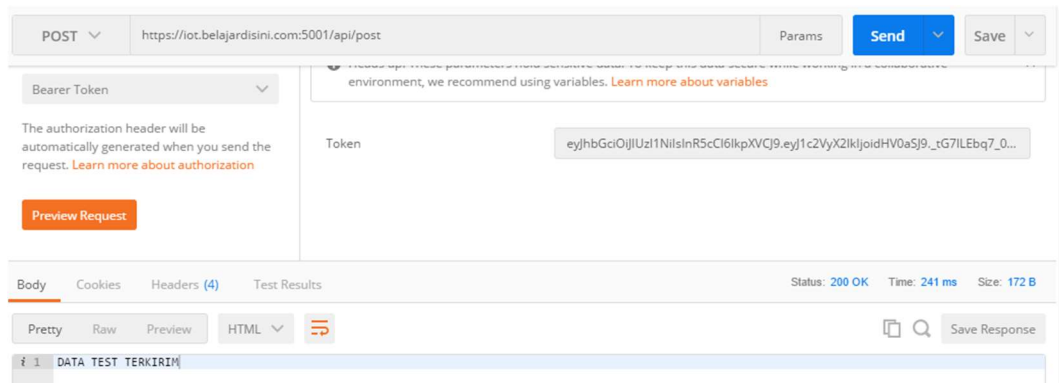
5.2.1.2 Methods *POST*

Pengimplementasian *webservice* dengan method *POST* bisa dilihat pada Tabel 5.3. Untuk melakukan *post*, terlebih dahulu diinisialisasi data *storage* yang akan digunakan. Ditambah lagi dengan pemberian kondisi untuk menyimpan data yang telah diterima dari *Internet Gateway Device* yang bermeta *text* dan gambar.

Tabel 5.4 Kode sumber pengimplementasian *post* data

Method : <i>post</i>	
1	@app.route('/api/post', methods=['POST'])
2	@token_required()
3	def post(self):
4	db = client.test
5	fs = gridfs.GridFS(db)
6	for data in request.get_json():
7	if "Data" in data or request.content_length >= (800*1024):
8	print(gridfs, data["Name"])
9	Gambar = pickle.loads(data["Data"].encode('utf-
10	8'), encoding='latin1')
11	fs.put(Gambar, filename=data["Name"],
12	encoding='latin1')
13	else :
14	db['testH_130'].insert_one(data)
15	return "DATA TEST TERKIRIM"

Hasil dari kode program implementasi *post* data dapat dilihat pada Gambar 5.10. dan Gambar 5.12 menunjukkan penggunaan token untuk melakukan *post* data dan keluaran program berupa tulisan "Data Test Terkirim".



Gambar 5.12 Output kode program post data

5.2.2 Implementasi JSON Web Token

Pengimplementasian JSON Web Token dibuat untuk membatasi hak akses data *storage* sehingga hanya pengguna yang memiliki token yang mampu mengakses data *storage*. Token bisa didapatkan dengan melakukan *login* terlebih dahulu dengan menggunakan kode program seperti pada Tabel 5.4 . Proses *login* akan diawali dengan pengecekan *username* dan *password* yang telah tersimpan pada database user kemudian pemberian token yang tersusun atas data-data user yang telah dienkripsi. Token dihasilkan dengan melewati proses encoding dan decoding yang dilakukan oleh library JWT.

Tabel 5.5 Kode sumber pengimplementasian token

Method : token_required	
1	def token_required(is_admin=False):
2	def token_required_inner(f):
3	@wraps(f)
4	def decorated(*args, **kwargs):
5	res = validate_token(request, is_admin)
6	if not res.get('user'):
7	return jsonify(res.get('message')), 401
8	return f(res.get('user'), *args, **kwargs)
9	return decorated
10	return token_required_inner

Setelah mendapatkan token, untuk melakukan *get* atau *post* data, token akan divalidasi terlebih dahulu menggunakan program seperti pada Tabel 5.5 . Validasi token dilakukan untuk memastikan lagi apakah yang akan mengakses data *storage* benar-benar telah terdaftar pada database user atau belum.

Tabel 5.6 Kode sumber validasi token

Method : validate_token	
1	def validate_token(request, is_admin=False):
2	
3	auth_header = request.headers.get('Authorization')
4	
5	if not auth_header or 'Bearer' not in auth_header:
6	return {'message': 'Bad authorization header!'}
7	
8	split = auth_header.split(' ')

```

9
10     try:
11         decode_data = jwt.decode(split[1],
12 app.config['SECRET_KEY'])
13         user =
14 User.query.filter_by(mac_address=decode_data.get('user_id')).first
15 ()
16
17         if not user:
18             return {'message': 'User not found'}
19         if is_admin and not user.is_admin:
20             return {'message': 'Not admin token'}
21
22         return {'user': user.as_dict()}
23
24     except Exception as error:
25         return {'message': 'Token is invalid'}

```

5.2.3 Implementasi Konsep *Semantic Interoperability*

Pengimplementasian konsep *Semantic Interoperability* ditunjukkan pada kode program pada Tabel 5.6 . Konsep *semantic interoperability* diterapkan dengan diterimanya meta data yang beragam dan berbagai macam ukuran data dari Internet *Gateway Device* yang kemudian akan disimpan pada data *storage* . Proses penyimpanan data pada data *storage* dilakukan sesuai dengan meta data dan ukuran data. Untuk meta data gambar dan data berukuran lebih dari sama dengan 800 kB akan disimpan pada Grid FS, sedangkan meta data *text* dan ukuran data kurang dari 800 kB akan disimpan pada Mongo DB.

Tabel 5.7 Kode sumber pengimplementasian konsep *semantic interoperability*

Method : <i>get</i> yang menampilkan konsep <i>semantic interoperability</i>	
1	<code>if request.args.get('topic') :</code>
2	<code> if request.args.get('topic')== "Gambar":</code>
3	<code> fs = gridfs.GridFS(db_)</code>
4	<code> for data_ in fs.find():</code>
5	
6	<code>data.append("https://iot.bermainmainditaman.com:5001/api/getGambar</code>
7	<code>/"+str(data_._id))</code>
8	<code> else:</code>
9	<code> for data_ in db_.testH__130.find({"topic":</code>
10	<code>request.args.get('topic')},{ "_id":0}):</code>
11	<code> data.append(data_)</code>
12	<code> else:</code>
13	<code> for data_ in db_.testH__130.find({}, {"_id": 0}):</code>
14	<code> data.append(data_)</code>
15	<code> return json.dumps(data, sort_keys=True)</code>

5.2.4 Implementasi *IoT Application*

Pengimplementasian *IoT Apps* berupa pembuatan tampilan web sederhana menggunakan bahasa HTML. Implementasi *IoT Application* diawali dengan membuat kode program yang mampu membaca data pada data *storage* dan mampu terhubung dengan kode program web *service*. Tampilan awal berupa halaman register yang ditunjukkan seperti pada Gambar 5.13 . Melalui proses

pembuatan akun terlebih dahulu untuk bisa melakukan *login* seperti pada Gambar 5.14.

REGISTER

Username :

Password :

Mac Address :

Gambar 5.13 Tampilan halaman *register*

LOGIN

Username :

Password :

☐ Remember me

Gambar 5.14 Tampilan halaman *login*

Gambar 5.15 merupakan tampilan halaman utama setelah pengguna melakukan *login*. Token akan tampil secara otomatis di kolom Input Token dan tersedia pula pilihan meta data yang bisa dipilih.

API Webservice For IoT Home

Sign Up Login Logout

SOP POST

1. Lakukan login terlebih dahulu untuk mendapatkan token
2. Silahkan gunakan salah satu API Tester Untuk mengakses fungsi post
3. Silahkan masukkan path pada kolom skema yang telah disediakan
4. Tambahkan header dengan nama header Authorization dan value Bearer (token yang didapatkan setelah login)
5. Masukkan data yang mau di post pada kolom Body
6. Pilih method POST
7. Klik tombol Send

SOP GET

1. Lakukan login terlebih dahulu untuk mendapatkan token
2. Silahkan gunakan salah satu API Tester Untuk mengakses fungsi GET
3. Silahkan masukkan path pada kolom skema yang telah disediakan
4. Tambahkan header dengan nama header Authorization dan value Bearer (token yang didapatkan setelah login)

Input Token :

☐ Text
☒ Gambar

Hasil Get Data

Response Method

Method	Jenis Data	Description
Get	Text	Menampilkan hasil sensor berupa humidity, protocol, sensor, temperatur, timestamp, dan topic
Get	Gambar	Menampilkan hasil sensor kamera berupa link gambar
Post	Text	Menginput data text
Post	Gambar	Menginput data gambar

Gambar 5.15 Tampilan halaman utama *IoT Application*

BAB 6 PENGUJIAN

Pada bab pengujian akan dilakukan beberapa uji coba terhadap apa yang telah diimplementasikan. Pengujian ini dilakukan sesuai dengan perancangan pengujian yang telah dijelaskan pada bab-bab sebelumnya. Adapun pengujian yang akan dilakukan yaitu pengujian fungsionalitas, pengujian skalabilitas, dan pengujian kinerja web *service*.

6.1 Pengujian Fungsionalitas

Pengujian fungsional dilakukan agar dapat mengetahui apakah fungsi-fungsi hasil implementasi dengan perancangan sesuai atau tidak. Dari hasil pengujian fungsionalitas, dapat diketahui apakah sistem itu berjalan dengan benar atau tidak.

6.1.1 Menerima Data *Text/JSON* Dari Internet *Gateway Device*

Kasus uji proses menerima data *text/JSON* dari Internet *Gateway Device* ditunjukkan pada Tabel 6.1

Tabel 6.1 Menerima data *text/JSON* dari IGD

Kode	WS_001
Nama Kasus Uji	Menerima data <i>text/JSON</i> dari Internet <i>Gateway Device</i>
Tujuan Pengujian	Mengetahui apakah web <i>service</i> bisa menerima data <i>text/JSON</i>
Prosedur Pengujian	<ol style="list-style-type: none"> 1. <i>Middleware</i> dijalankan 2. Pengguna menjalankan IGD 3. Pengguna melihat data <i>text/JSON</i> yang masuk melalui Putty
Hasil yang Diharapkan	Web <i>service</i> dapat menerima data <i>text/JSON</i> dari Internet <i>Gateway Device</i>
Hasil Pengujian	Web <i>service</i> dapat menerima data <i>text/JSON</i> dari Internet <i>Gateway Device</i>

Gambar 6.1 menunjukkan bahwa web *service* berhasil menerima data *text/JSON* dari Internet *Gateway Device* (IGD). Pada Gambar 6.1 menampilkan alamat ip dari Internet *Gateway Device*, waktu dilakukannya penerimaan data, beserta isi dari data yang diterima dari Internet *Gateway Device*. Penerimaan data

```
175.45.189.36 - - [20/May/2018 09:07:38] "POST /login HTTP/1.1" 200 -
[{"topic": "office/roomA14", "temperature": {"unit": "celcius", "value": "26.0"},
, "timestamp": 0, "sensor": {"ip": "192.168.42.12", "tipe": "esp8266", "index":
16446677, "module": "dht22"}, "humidity": {"unit": "%", "value": "72.0"}, "proto
col": "coap"}]
175.45.189.36 - - [20/May/2018 09:07:38] "POST /api/post HTTP/1.1" 200 -
```

Gambar 6.1 Menerima data text/JSON dari IGD

dari Internet *Gateway Device* berhasil dibuktikan dengan adanya kode respon 200 yang berarti OK.

6.1.2 Menerima Data Gambar Dari Internet *Gateway Device*

Kasus uji proses menerima data gambar dari Internet *Gateway Device* ditunjukkan pada Tabel 6.2

Tabel 6.2 Menerima data gambar dari IGD

Kode	WS_002
Nama Kasus Uji	Menerima data gambar dari Internet <i>Gateway Device</i>
Tujuan Pengujian	Mengetahui apakah web <i>service</i> bisa menerima data gambar
Prosedur Pengujian	<ol style="list-style-type: none"> 1. <i>Middleware</i> dijalankan 2. Pengguna menjalankan IGD 3. Pengguna melihat data gambar yang masuk melalui Putty
Hasil yang Diharapkan	Web <i>service</i> dapat menerima data gambar dari Internet <i>Gateway Device</i>
Hasil Pengujian	Web <i>service</i> dapat menerima data gambar dari Internet <i>Gateway Device</i>

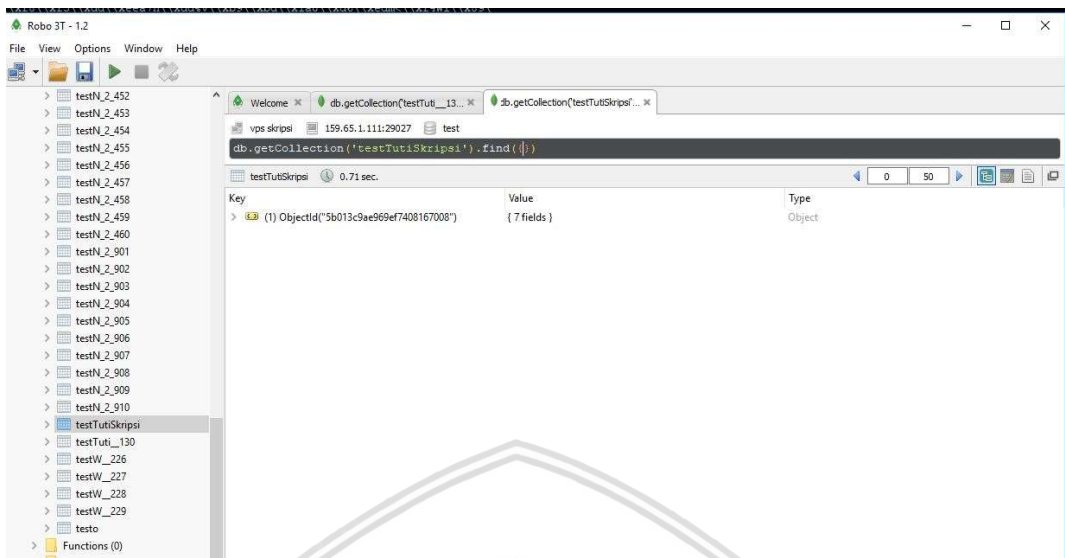
Gambar 6.2 Menerima data Gambar dari IGD

6.1.3 Menyimpan Data Dari Internet Gateway Device ke Data Storage

Tabel 6.3 Menyimpan data dari IGD ke data *storage*

37

Gambar 6.3 menunjukkan bahwa data dari Internet Gateway Device (IGD)



Gambar 6.3 Menyimpan data dari IGD ke data storage

bisa tersimpan di dalam data storage. Data yang telah tersimpan pada data storage dapat dilihat pada RoboMongo. Data akan tersimpan dalam function yang telah dituliskan pada kode program. Dalam function tersebut, terdapat deskripsi terkait data apa saja yang telah tersimpan beserta waktu berlangsungnya penyimpanan data.

6.1.4 Melakukan Autentikasi Dan Otorisasi Terhadap Subscriber Yang Akan Mengambil Data

Kasus uji autentikasi dan otorisasi terhadap subscriber yang akan mengambil data ditunjukkan pada Tabel 6.4.

Tabel 6.4 Uji autentikasi dan otorisasi

Kode	WS_004
Nama Kasus Uji	Melakukan autentikasi dan otorisasi terhadap subscriber yang akan mengambil data.
Tujuan Pengujian	Mengetahui apakah web service bisa melakukan autentikasi dan otorisasi terhadap subscriber yang akan mengambil data.
Prosedur Pengujian	1. Middleware dijalankan. 2. Pengguna menjalankan IGD. 3. Pengguna melakukan pendaftaran dengan memasukkan username, password dan mac address dari device yang digunakan. 4. Pengguna melakukan login menggunakan username dan password untuk mendapatkan token.

Hasil yang Diharapkan	Web <i>service</i> dapat melakukan autentikasi dan otorisasi terhadap <i>subscriber</i> yang akan mengambil data.
Hasil Pengujian	Web <i>service</i> dapat melakukan autentikasi dan otorisasi terhadap <i>subscriber</i> yang akan mengambil data.



Gambar 6.4 Proses login gagal

Autentikasi pada web *service* dapat ditunjukkan dengan adanya verifikasi *username* dan *password* pada saat *login*, serta pemberian token saat *login* telah berhasil dilakukan. Pengguna yang telah mendaftarkan *username*, *password*, dan *mac address* akan mempunyai hak akses untuk *login*. Namun, jika pengguna tidak mendaftarkan *username*, *password*, dan *mac address* maka pengguna tidak akan mendapatkan hak akses. Gambar 6.4 menunjukkan kondisi jika pengguna memasukkan *username* atau *password* yang tidak terdaftar. Akan tampil pesan error yang menuliskan “*message : invalid login*”.

Setelah pengguna berhasil melakukan *login* maka pengguna akan mendapatkan token untuk melakukan fungsi *get* dan *post* pada web *service* seperti yang ditunjukkan pada Gambar 6.5. Pada saat pengguna berhasil login, akan tampil halaman utama di mana pengguna bisa melakukan fungsi *get* data dengan menggunakan token yang telah didapatkan.

API Webservice For IoT
 Home
 Sign Up
 Login
 Logout

SOP POST

1. Lakukan login terlebih dahulu untuk mendapatkan token
2. Silahkan gunakan salah satu API Tester Untuk mengakses fungsi post
3. Silahkan masukkan path pada kolom skema yang telah disediakan
4. Tambahkan header dengan nama header **Authorization** dan value **Bearer (token yang didapatkan setelah login)**
5. Masukkan data yang mau di post pada kolom Body
6. Pilih method **POST**
7. Klik tombol **Send**

SOP GET

1. Lakukan login terlebih dahulu untuk mendapatkan token
2. Silahkan gunakan salah satu API Tester Untuk mengakses fungsi GET
3. Silahkan masukkan path pada kolom skema yang telah disediakan
4. Tambahkan header dengan nama header **Authorization** dan value **Bearer (token**

Input Token :

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkjoidHV0aSJ9.1G7iLEBq7_0nuQUr-FReg4K-ALPa6Jb8OOTuXALVy0

☐ Text
☒ Gambar

GET

Hasil Get Data

Response Method

Method	Jenis Data	Description
Get	Text	Menampilkan hasil sensor berupa humidity, protocol, sensor, temperatur, timestamp, dan topic
Get	Gambar	Menampilkan hasil sensor kamera berupa link gambar
Post	Text	Menginput data text
Post	Gambar	Menginput data gambar

Gambar 6.5 Proses login berhasil

Penyediaan akses kontrol akan secara otomatis didapatkan oleh pengguna setelah mendapatkan token. Namun token yang didapatkan oleh pengguna juga memiliki masa aktif sehingga mengharuskan pengguna untuk terus memperbaharui token dalam kurun waktu tertentu. Gambar 6.6 menunjukkan jika token telah kadaluarsa atau sudah tidak berlaku. Dalam hal ini, pengguna harus melakukan *login* kembali untuk mendapatkan token ketika ingin melakukan akses data dalam *data storage*. *Login* dapat dilakukan dengan menginputkan kembali *username* dan *password* yang telah didaftarkan sehingga token dapat didapatkan kembali. Pada Gambar 6.6 terdapat pesan “*Authentication Failed*” yang menandakan bahwa autentikasi gagal dilakukan karena token sudah tidak berlaku lagi.

Input Token :

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkjoidHV0aSJ9.1G7iLEBq7_0nuQUr-FReg4K-ALPa6Jb8OOTuXALVy0

☐ Text
☒ Gambar

GET

Hasil Get Data

```

b<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<title>pymongo.errors.OperationFailure: Authentication failed. // Werkzeug Debugger</title>
<link rel="stylesheet" href="?"
__debugger__=yes&cmd=resource&f=style.css" type="text/css">
<!-- We need to make sure this has a favicon so that the debugger
does not by accident trigger a request to /favicon.ico which might
change the application state. -->
<link rel="shortcut icon"
href="?"__debugger__=yes&cmd=resource&f=console.png">
<script src="?"__debugger__=yes&cmd=resource&f=jquery.js">
</script>
<script src="?"__debugger__=yes&cmd=resource&f=debugger.js">
</script>
<script type="text/javascript">
var
TRACEBACK = 140088717657816,
CONSOLE_MODE = false,
EVALEX = true,
EVALEX_TRUSTED = false,
SECRET =
"HCxNrQadceFY98naCPHB";
</script>
</head>
<body style="background-color: #fff">
<div
class="debugger">
<h1>pymongo.errors.OperationFailure</h1>
<div class="detail">
<p class="errorMsg">pymongo.errors.OperationFailure:
Authentication failed.
</div>
</div>
</div>
                
```

Gambar 6.6 Penggunaan token yang kadaluarsa

Biasanya, terdapat kesalahan penulisan token ataupun *human error* yang bisa menimbulkan kesalahan dalam penulisan token. Penggunaan token yang tidak tepat akan menunjukkan pesan *error* seperti pada Gambar 6.7 . Akan muncul pesan *error* yang bertuliskan “*Token si invalid*” yang menandakan bahwa token

yang digunakan salah atau tidak valid sehingga pengguna harus melakukan *login* kembali untuk mendapatkan token yang benar dan valid.

Input Token :

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoiaHV0aSJ9._tG7ILEbq7_0nuQUr-FReg4K-ALPa6Jb8OOTuXALVy0

☐ Text
☒ Gambar

GET

Hasil Get Data

b"Token is invalid"\n'

Gambar 6.7 Penggunaan token yang tidak tepat

Dengan memasukkan token yang tepat, pengguna akan mendapat hak akses untuk melakukan fungsi *get* data dan *post* data seperti yang ditunjukkan pada Gambar 6.8 dan Gambar 6.9 . Ketika pengguna telah mendapatkan hak akses, pengguna bisa melakukan penyimpanan data dalam *data storage* dan bisa melakukan pengambilan data. Pada Gambar 6.8 merupakan tampilan di mana pengguna telah berhasil melakukan salah satu fungsi *get* meta data yaitu *get* meta data gambar. Hasil dari *get* data gambar yaitu berupa *link website* yang bisa diklik ketika ingin melihat Gambar yang dihasilkan.

Input Token :

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoiaHV0aSJ9._tG7ILEbq7_0nuQUr-FReg4K-ALPa6Jb8OOTuXALVy0

☐ Text
☒ Gambar

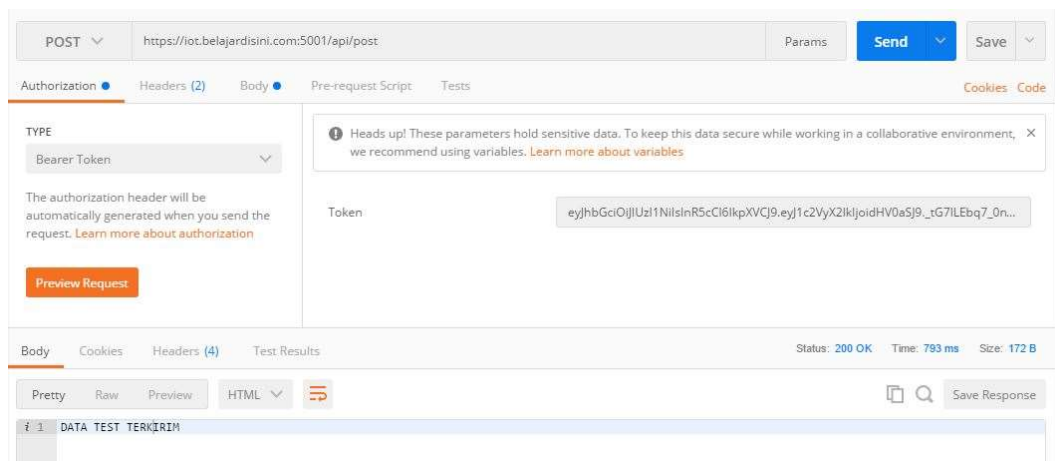
GET

Hasil Get Data

b["https://iot.belajardisini.com:5001/api/getgambar/5ae03da4e969ef09e11879d2",
"https://iot.belajardisini.com:5001/api/getgambar/5ae03da4e969ef09e11879e2",
"https://iot.belajardisini.com:5001/api/getgambar/5ae03da4e969ef09e11879e8",
"https://iot.belajardisini.com:5001/api/getgambar/5ae12debe969ef1c572a5b37",
"https://iot.belajardisini.com:5001/api/getgambar/5ae1327ae969ef1c572a5b3d",
"https://iot.belajardisini.com:5001/api/getgambar/5ae13df4e969ef1c572a5b44",
"https://iot.belajardisini.com:5001/api/getgambar/5ae14282e969ef1c572a5b4a",
"https://iot.belajardisini.com:5001/api/getgambar/5ae14822e969ef1c572a5b4e",
"https://iot.belajardisini.com:5001/api/getgambar/5ae14c18e969ef1c572a5b50",
"https://iot.belajardisini.com:5001/api/getgambar/5ae15453e969ef1c572a5b52",
"https://iot.belajardisini.com:5001/api/getgambar/5ae56fbce969ef258d8b5427",
"https://iot.belajardisini.com:5001/api/getgambar/5ae56fbce969ef258d8b542d"]

Gambar 6.8 Berhasil menggunakan token untuk melakukan fungsi *get* data

Pada Gambar 6.9 merupakan tampilan di mana pengguna berhasil melakukan *post* data atau melakukan penyimpanan data. Hal ini ditandai dengan adanya pesan berhasil "Data Test Terkirim".



Gambar 6.9 Berhasil menggunakan token untuk melakukan fungsi *post* data

6.1.5 Menyimpan Data dan Mengenali Meta Data Yang Akan disimpan ke Dalam Data *Storage*

Kasus uji menyimpan data dan mengenali meta data yang akan disimpan ke dalam data *storage* ditunjukkan pada Tabel 6.5.

Tabel 6.5 Fungsi menyimpan dan mengenali data

Kode	WS_005
Nama Kasus Uji	Menyimpan data dan mengenali meta data yang akan disimpan ke dalam data <i>storage</i> .
Tujuan Pengujian	Mengetahui apakah web <i>service</i> bisa menyimpan data dan mengenali meta data yang akan disimpan ke dalam data <i>storage</i> .
Prosedur Pengujian	<ol style="list-style-type: none"> 1. <i>Middleware</i> dijalankan. 2. Pengguna menjalankan IGD. 3. Pengguna menggunakan token yang telah didapatkan setelah <i>login</i> untuk melakukan penyimpanan data sesuai dengan ukuran dan meta data. 4. Data akan dipisahkan tempat penyimpanan datanya sesuai dengan meta data dan ukuran datanya.
Hasil yang Diharapkan	Web <i>service</i> dapat mengetahui apakah web <i>service</i> bisa menyimpan data dan mengenali meta data yang akan disimpan ke dalam data <i>storage</i> .
Hasil Pengujian	Web <i>service</i> dapat mengetahui apakah web <i>service</i> bisa menyimpan data dan mengenali meta data yang akan disimpan ke dalam data <i>storage</i> .

Mengenali meta data merupakan salah satu bentuk implementasi konsep *semantic interoperability* dengan menyimpan data sesuai dengan ukuran data dan meta data nya. Untuk data berukuran minimal 800 kB dan mempunyai meta data gambar akan disimpan pada MongoDB seperti Gambar 6.10.

Pada Gambar 6.10 terdapat alamat ip dari pengguna yang melakukan penyimpanan data beserta dengan waktu dilakukannya penyimpanan data tersebut. Kemudian status menampilkan pesan yang berisikan data tersebut disimpan di data *storage* yang mana sesuai dengan meta data dan ukuran data yang disimpan tadi. Berhasilnya penyimpanan data ditandai dengan adanya kode respons 200.

```
66.96.233.11 - - [20/Jun/2018 13:22:29] "POST /home HTTP/1.1" 200 -
Data Text Tersimpan di MongoDB
```

Gambar 6.10 Data berhasil disimpan di MongoDB

Sedangkan data berukuran kurang dari 800 kB dan mempunyai meta data *text* akan disimpan pada GridFS seperti pada Gambar 6.11 . Pada Gambar 6.11 menampilkan hasil keluaran yaitu berupa nama file gambar yang disimpan, alamat ip dari pengguna yang melakukan penyimpanan gambar, waktu dilakukannya penyimpanan gambar, dan pesan yang menjelaskan di data *storage* mana data gambar tersebut tersimpan. Kode respon 200 menandakan bahwa proses penyimpanan data ini berhasil dilakukan.

```
images0.jpg
Data Gambar Tersimpan di Grid FS
66.96.233.11 - - [20/Jun/2018 13:51:44] "POST /api/post HTTP/1.1" 200 -
```

Gambar 6.11 Data berhasil disimpan di GridFS

6.1.6 Melakukan fungsi *POST* dan *GET* data

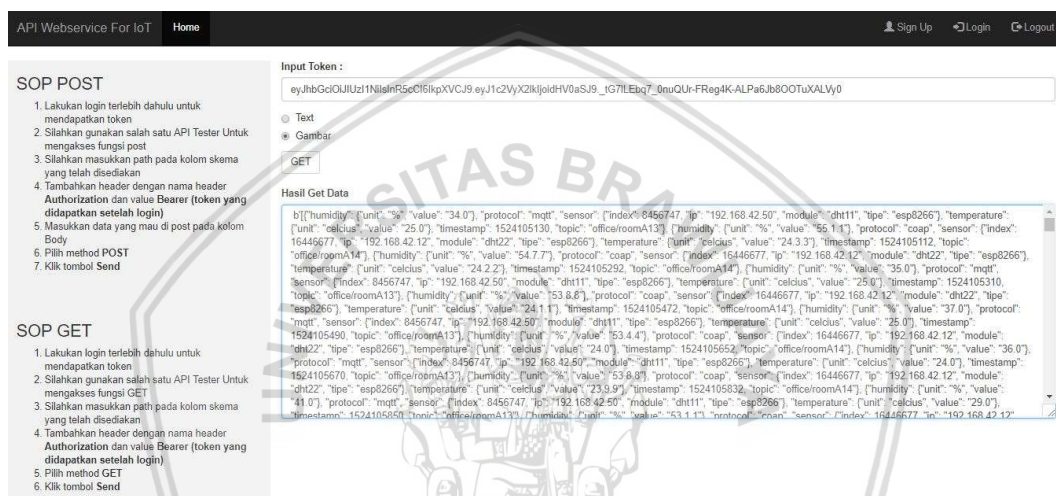
Kasus uji melakukan fungsi *POST* dan *GET* data ditunjukkan pada Tabel 6.6

Tabel 6.6 Melakukan fungsi *POST* dan *GET* data

Kode	WS_006
Nama Kasus Uji	Melakukan fungsi <i>POST</i> dan <i>GET</i> data.
Tujuan Pengujian	Mengetahui apakah web <i>service</i> bisa melakukan fungsi <i>POST</i> dan <i>GET</i> data.
Prosedur Pengujian	<ol style="list-style-type: none"> 1. <i>Middleware</i> dijalankan. 2. Pengguna menjalankan IGD. 3. Pengguna melakukan <i>login</i> menggunakan <i>username</i>, <i>password</i>, dan <i>mac address</i> yang telah didaftarkan untuk mendapatkan token. 4. Pengguna mengambil dan menyimpan data (melakukan fungsi <i>get</i> dan <i>post</i>) menggunakan token yang telah didapatkan.

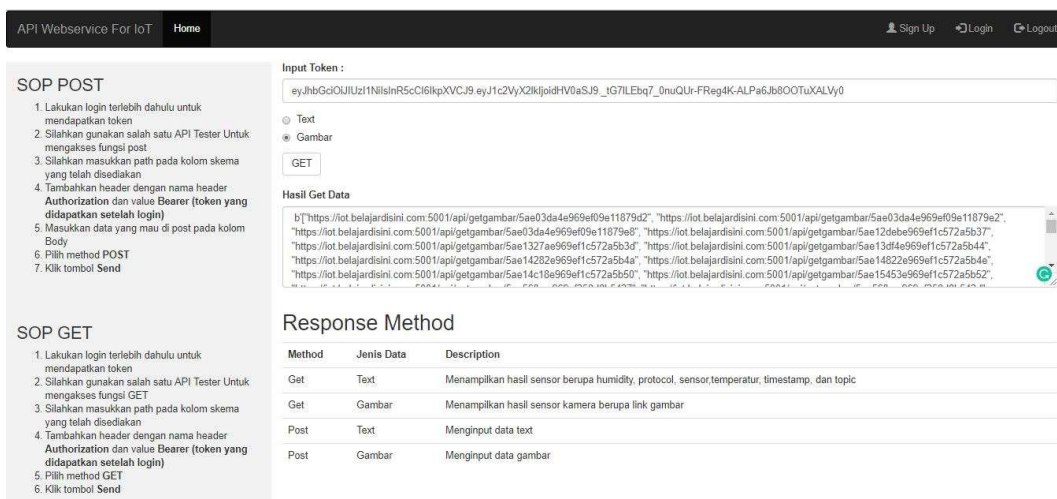
Hasil yang Diharapkan	Web service dapat mengetahui apakah web service bisa melakukan fungsi <i>POST</i> dan <i>GET</i> data.
Hasil Pengujian	Web service dapat mengetahui apakah web service bisa melakukan fungsi <i>POST</i> dan <i>GET</i> data.

Kemampuan melakukan fungsi *get* dan *post* data menjadi fungsionalitas utama pada sebuah web service. Pada pengujian fungsionalitas ini, akan dilihat apakah web service mampu melakukan fungsi *get* dan *post* data. Pada Gambar 6.12 menunjukkan bahwa web service mampu melakukan *get* data *text* dengan tampilnya data-data *text* pada kolom hasil *get*. Pada kolom hasil *get* data ditampilkan data yang telah didapatkan dari sensor yang berisikan data suhu dan data kelembapan.



Gambar 6.12 Hasil *get* data text

Pada Gambar 6.13 menunjukkan bahwa web service mampu melakukan *get* data gambar yang ditunjukkan dengan tampilnya URL pada kolom hasil *get* data yang berisikan data gambar dari sensor Gambar. Sedangkan pada Gambar 6.14 menunjukkan Gambar hasil dari pengaksesan salah satu URL dari kolom *get* data pada Gambar 6.13. Gambar yang dihasilkan berupa Gambar langit-langit ruangan diletakkannya sensor Gambar.

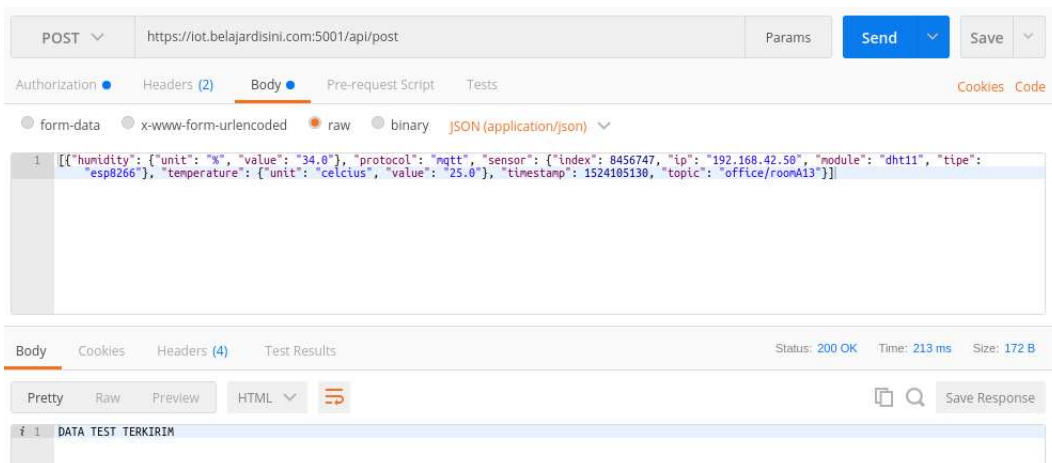


Gambar 6.13 Hasil *get* data gambar

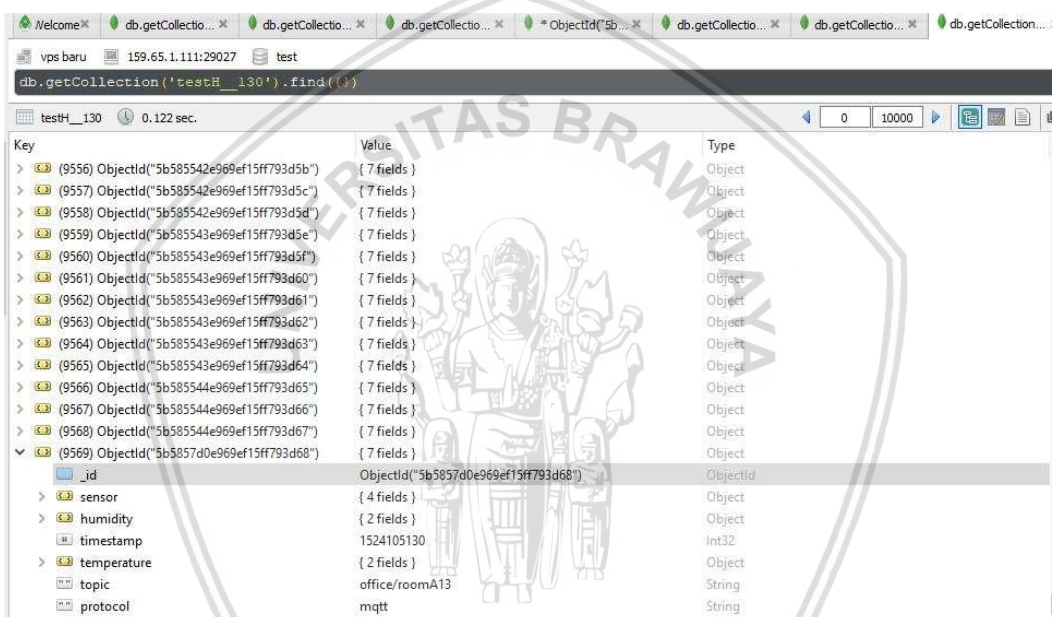


Gambar 6.14 Salah satu data gambar yang diambil

Selanjutnya untuk kemampuan penyimpanan data atau *post* data dapat dilihat pada Gambar 6.15 dan 6.17 . *Post* data dilakukan menggunakan *tool* Postman. Gambar 6.15 terdapat kolom *body* yang berisikan data text yang ingin disimpan dan di kolom *result* menampilkan pesan “Data Test Terkirim” yang menandakan bahwa data yang dituliskan tadi telah tersimpan. Bukti bahwa data text telah tersimpan dalam data *storage* dapat dilihat menggunakan *tol* RoboMongo seperti pada Gambar 6.16 .

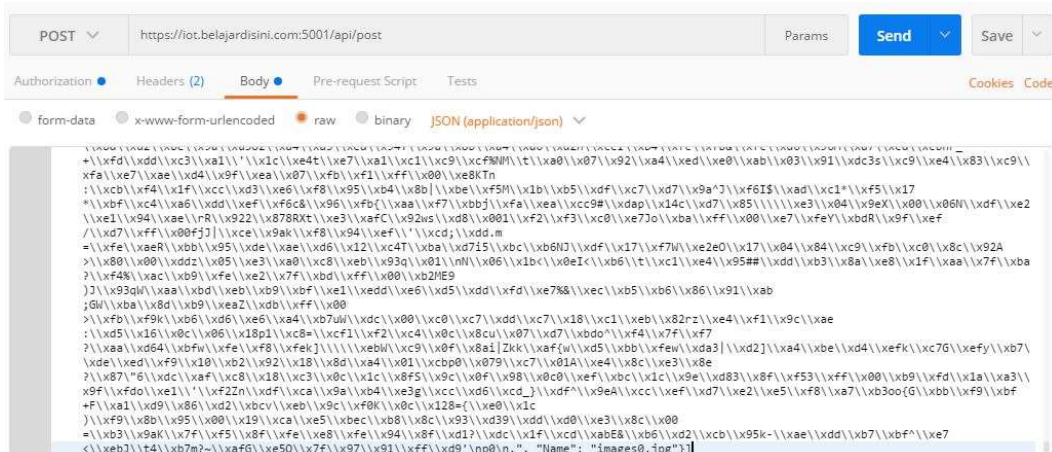


Gambar 6.15 Berhasil *post* data text



Gambar 6.16 Data text telah tersimpan dalam data *storage*

Pada Gambar 6.17 menunjukkan kolom *body* pada *tool* Postman yang berisikan data gambar yang ingin disimpan pada data *storage* . Gambar 6.18 menunjukkan pesan "Data Test Terkirim" yang menunjukkan bahwa data tersebut telah disimpan dalam data *storage* .



Gambar 6.17 Data Gambar yang akan disimpan



Gambar 6.18 Berhasil *post* data gambar

Data Gambar yang telah tersimpan, dapat dilihat pada *tool* RoboMongo seperti pada Gambar 6.19 yang memperlihatkan bahwa data gambar yang tadi *dipost* telah tersimpan dalam data *storage* . Pada Gambar 6.19 menunjukkan *id* dari data gambar, ukuran dari data gambar, waktu saat penyimpanan data dilakukan, juga nama *file* dari data gambar yang tersimpan.

Key	Value	Type
> (29) ObjectId("5ae951e2e969ef6fdbc5cdc6")	{ 6 fields }	Object
> (30) ObjectId("5ae951e2e969ef6fdbc5cdd0")	{ 6 fields }	Object
> (31) ObjectId("5ae951e2e969ef6fdbc5cdd9")	{ 6 fields }	Object
> (32) ObjectId("5ae951e2e969ef6fdbc5cde5")	{ 6 fields }	Object
> (33) ObjectId("5ae951e2e969ef6fdbc5cde9")	{ 6 fields }	Object
> (34) ObjectId("5ae97f24e969ef6fdbc5cdfc")	{ 6 fields }	Object
> (35) ObjectId("5ae97f85e969ef6fdbc5ce03")	{ 6 fields }	Object
> (36) ObjectId("5ae97f95e969ef6fdbc5ce06")	{ 6 fields }	Object
> (37) ObjectId("5b013b26e969ef739faec01")	{ 7 fields }	Object
> (38) ObjectId("5b08d49de969ef7d2a386af3")	{ 6 fields }	Object
> (39) ObjectId("5b0917d9e969ef071a56d9ab")	{ 6 fields }	Object
> (40) ObjectId("5b0f725de969ef69440850e7")	{ 6 fields }	Object
> (41) ObjectId("5b2a5bf0e969ef3a5ca5c46d")	{ 7 fields }	Object
✓ (42) ObjectId("5b586083e969ef15ff793d69")	{ 7 fields }	Object
_id	ObjectId("5b586083e969ef15ff793d69")	ObjectId
chunkSize	261120	Int32
md5	4f9e608c72024c30802cd7c5388912f7	String
uploadDate	2018-07-25 11:35:31.555Z	Date
encoding	latin1	String
filename	images0.jpg	String
length	219897	Int32

Gambar 6.19 Data Gambar telah tersimpan dalam data *storage*

6.2 Pengujian Skalabilitas API Web Service

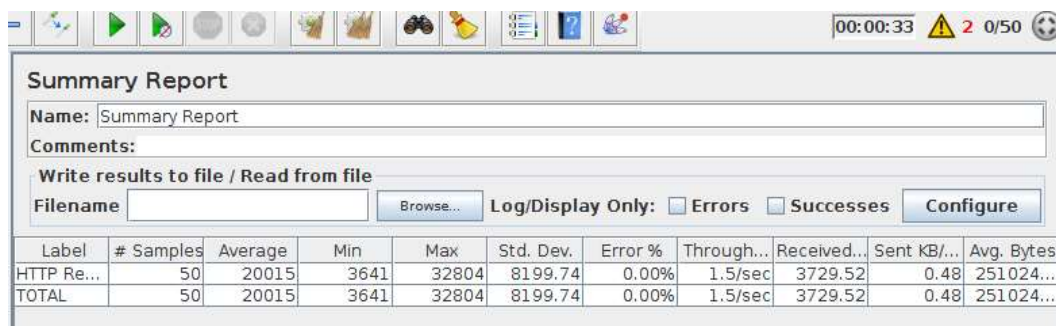
Pengujian skalabilitas dilakukan untuk mengetahui seberapa banyak data yang bisa dikirim dengan menggunakan fungsi *POST* dan seberapa banyak data yang bisa diambil dengan menggunakan fungsi *GET*. Pengujian ini menggunakan *tool* yaitu Apache JMeter.

6.2.1 Pengujian *GET* Data

Dalam pengujian ini, penguji mencoba melakukan pengambilan keseluruhan data yang ada pada data *storage*. Skenario yang diujikan dengan membuat 50,100,150,200,250,300 *request* data ke web *service*. Banyak data yang berhasil diambil beserta lamanya proses pengambilan data bergantung pada koneksi internet yang digunakan.

6.2.1.1 Skenario 1 : Melakukan *GET* data sebanyak 50 request data

Pada skenario ini, penguji mencoba melakukan *GET* data sebanyak 50 request dan didapatkan hasil seperti pada Gambar 6.20 yang menunjukkan persentase keberhasilan pengambilan data beserta waktu yang digunakan untuk mengambil data.



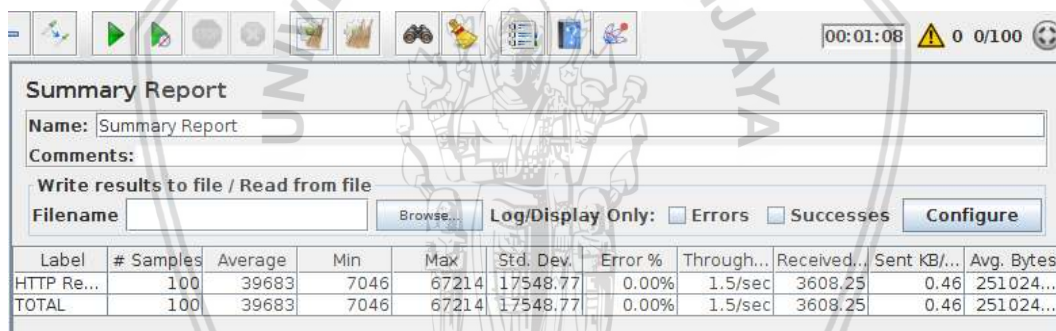
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Through...	Received...	Sent KB/...	Avg. Bytes
HTTP Re...	50	20015	3641	32804	8199.74	0.00%	1.5/sec	3729.52	0.48	251024...
TOTAL	50	20015	3641	32804	8199.74	0.00%	1.5/sec	3729.52	0.48	251024...

Gambar 6.20 Summary report test 50 request

Dari pengujian 50 request didapatkan hasil bahwa keseluruhan data berhasil diambil dengan persentase *error* 0% dengan menghabiskan waktu selama 33 detik.

6.2.1.2 Skenario 2 : Melakukan *GET* data sebanyak 100 request data

Pada skenario ini, penguji mencoba melakukan *GET* data sebanyak 100 request dan didapatkan hasil seperti pada Gambar 6.21 menunjukkan persentase keberhasilan pengambilan data beserta waktu yang digunakan untuk mengambil data.



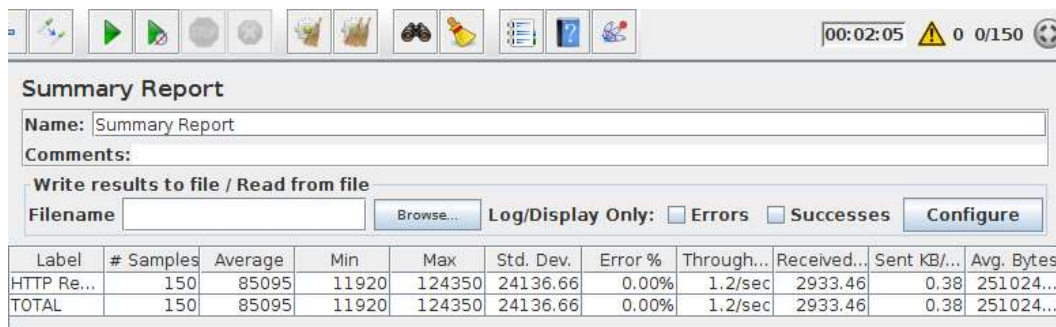
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Through...	Received...	Sent KB/...	Avg. Bytes
HTTP Re...	100	39683	7046	67214	17548.77	0.00%	1.5/sec	3608.25	0.46	251024...
TOTAL	100	39683	7046	67214	17548.77	0.00%	1.5/sec	3608.25	0.46	251024...

Gambar 6.21 Summary report test 100 request

Dari pengujian 100 request didapatkan hasil bahwa keseluruhan data berhasil diambil dengan persentase *error* 0% dengan menghabiskan waktu selama 68 detik.

6.2.1.3 Skenario 3 : Melakukan *GET* data sebanyak 150 request data

Pada skenario ini, penguji mencoba melakukan *GET* data sebanyak 150 request dan didapatkan hasil seperti pada Gambar 6.22 menunjukkan persentase keberhasilan pengambilan data beserta waktu yang digunakan untuk mengambil data.



Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename: Browse...

Log/Display Only: ☐ Errors ☐ Successes

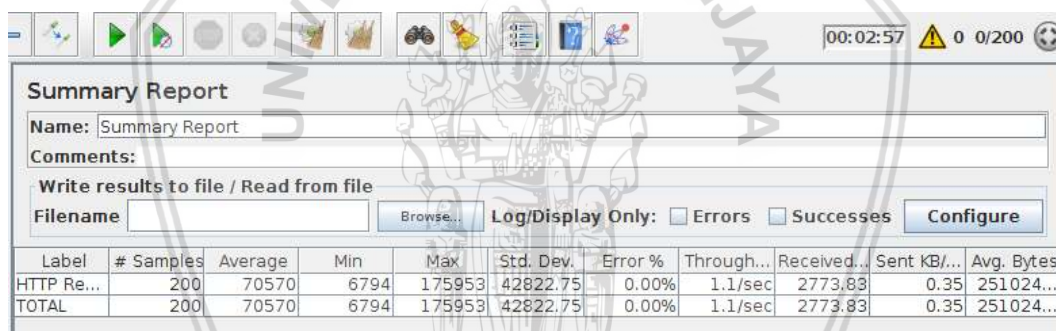
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Through...	Received...	Sent KB/...	Avg. Bytes
HTTP Re...	150	85095	11920	124350	24136.66	0.00%	1.2/sec	2933.46	0.38	251024...
TOTAL	150	85095	11920	124350	24136.66	0.00%	1.2/sec	2933.46	0.38	251024...

Gambar 6.22 Summary report test 150 request

Dari pengujian 150 request didapatkan hasil bahwa keseluruhan data berhasil diambil dengan persentase *error* 0% dengan menghabiskan waktu selama 125 detik.

6.2.1.4 Skenario 4 : Melakukan *GET* data sebanyak 200 request data

Pada skenario ini, penguji mencoba melakukan *GET* data sebanyak 200 request dan didapatkan hasil seperti pada Gambar 6.23 menunjukkan persentase keberhasilan pengambilan data beserta waktu yang digunakan untuk mengambil data.



Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename: Browse...

Log/Display Only: ☐ Errors ☐ Successes

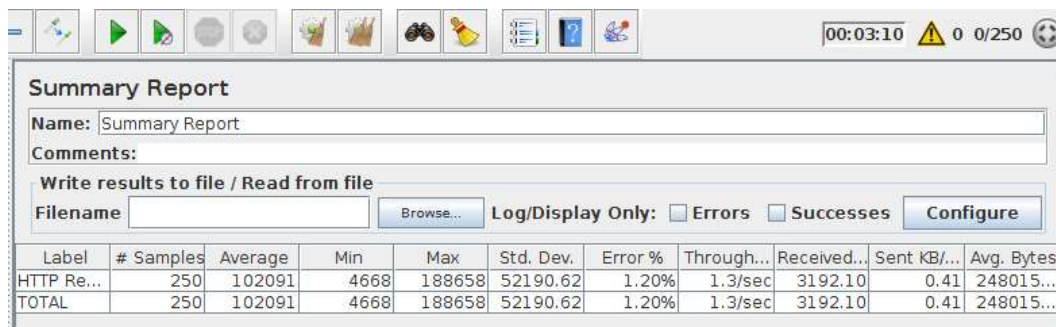
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Through...	Received...	Sent KB/...	Avg. Bytes
HTTP Re...	200	70570	6794	175953	42822.75	0.00%	1.1/sec	2773.83	0.35	251024...
TOTAL	200	70570	6794	175953	42822.75	0.00%	1.1/sec	2773.83	0.35	251024...

Gambar 6.23 Summary report test 200 request

Dari pengujian 200 request didapatkan hasil bahwa keseluruhan data berhasil diambil dengan persentase *error* 0% dengan menghabiskan waktu selama 177 detik.

6.2.1.5 Skenario 5 : Melakukan *GET* data sebanyak 250 request data

Pada skenario ini, penguji mencoba melakukan *GET* data sebanyak 250 request dan didapatkan hasil seperti pada Gambar 6.24 menunjukkan persentase keberhasilan pengambilan data beserta waktu yang digunakan untuk mengambil data.



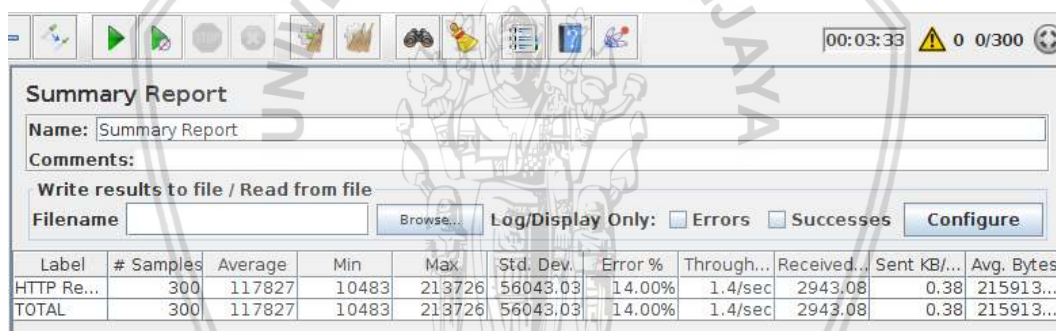
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Through...	Received...	Sent KB/...	Avg. Bytes
HTTP Re...	250	102091	4668	188658	52190.62	1.20%	1.3/sec	3192.10	0.41	248015...
TOTAL	250	102091	4668	188658	52190.62	1.20%	1.3/sec	3192.10	0.41	248015...

Gambar 6.24 Summary report test 250 request

Dalam pengujian 250 request didapatkan hasil bahwa data yang berhasil diambil yaitu 247 data dengan persentase *error* 1,20% dengan menghabiskan waktu selama 190 detik.

6.2.1.6 Skenario 6 : Melakukan *GET* data sebanyak 300 request data

Pada skenario ini, penguji mencoba melakukan *GET* data sebanyak 300 request dan didapatkan hasil seperti pada Gambar 6.25 menunjukkan persentase keberhasilan pengambilan data beserta waktu yang digunakan untuk mengambil data.



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Through...	Received...	Sent KB/...	Avg. Bytes
HTTP Re...	300	117827	10483	213726	56043.03	14.00%	1.4/sec	2943.08	0.38	215913...
TOTAL	300	117827	10483	213726	56043.03	14.00%	1.4/sec	2943.08	0.38	215913...

Gambar 6.25 Summary report test 300 request

Dari pengujian 300 request didapatkan hasil bahwa data yang berhasil diambil yaitu 258 data dengan persentase *error* 14% dengan menghabiskan waktu selama 213 detik.

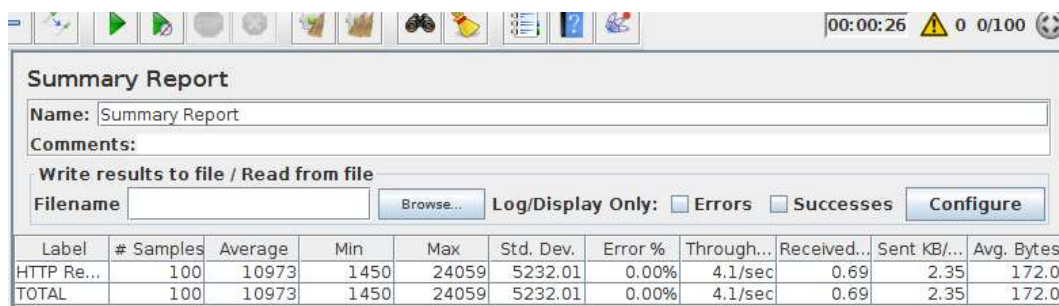
6.2.2 Pengujian *POST* Data

Dalam pengujian ini, penguji mencoba melakukan pengiriman data *text* ke dalam data *storage*. Skenario yang diujikan dengan membuat 100,300,500,700,1000 kali *POST* data ke web *service*. Banyak data yang berhasil dikirim beserta lamanya proses pengiriman data bergantung pada koneksi internet yang digunakan.

6.2.2.1 Skenario 1 : Melakukan *POST* data sebanyak 100 kali

Pada skenario ini, penguji mencoba melakukan *POST* data sebanyak 100 kali dan didapatkan hasil seperti pada Gambar 6.26 menunjukkan persentase

keberhasilan penyimpanan data beserta waktu yang digunakan untuk menyimpan data.



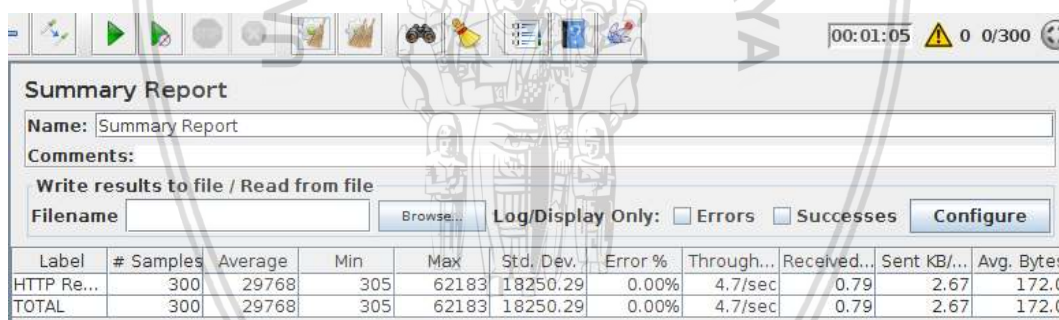
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Through...	Received...	Sent KB/...	Avg. Bytes
HTTP Re...	100	10973	1450	24059	5232.01	0.00%	4.1/sec	0.69	2.35	172.0
TOTAL	100	10973	1450	24059	5232.01	0.00%	4.1/sec	0.69	2.35	172.0

Gambar 6.26 Summary report test 100 kali

Dari pengujian 100 kali pengiriman data didapatkan hasil bahwa keseluruhan data berhasil dikirim dengan persentase *error* 0% dengan menghabiskan waktu selama 26 detik.

6.2.2.2 Skenario 2 : Melakukan *POST* data sebanyak 300 kali

Pada skenario ini, penguji mencoba melakukan *POST* data sebanyak 300 kali dan didapatkan hasil seperti pada Gambar 6.27 menunjukkan persentase keberhasilan penyimpanan data beserta waktu yang digunakan untuk menyimpan data.



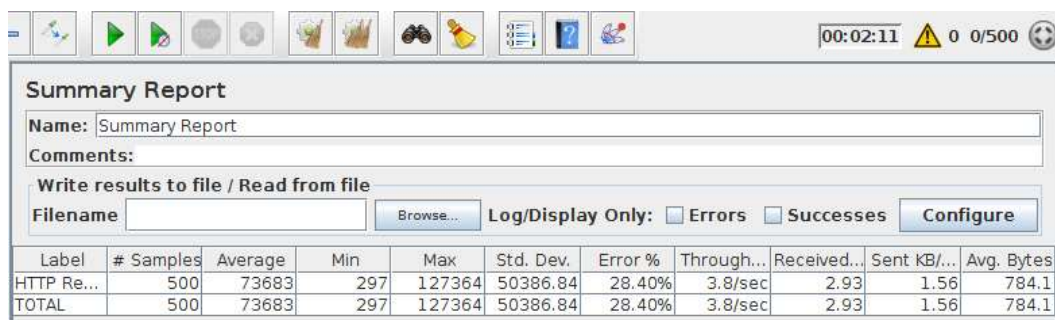
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Through...	Received...	Sent KB/...	Avg. Bytes
HTTP Re...	300	29768	305	62183	18250.29	0.00%	4.7/sec	0.79	2.67	172.0
TOTAL	300	29768	305	62183	18250.29	0.00%	4.7/sec	0.79	2.67	172.0

Gambar 6.27 Summary report test 300 kali

Dari pengujian 300 kali didapatkan hasil bahwa keseluruhan data berhasil dikirim dengan persentase *error* 0% dengan menghabiskan waktu selama 65 detik.

6.2.2.3 Skenario 3 : Melakukan *POST* data sebanyak 500 kali

Pada skenario ini, penguji mencoba melakukan *POST* data sebanyak 500 kali dan didapatkan hasil seperti pada Gambar 6.28 menunjukkan persentase keberhasilan penyimpanan data beserta waktu yang digunakan untuk menyimpan data.



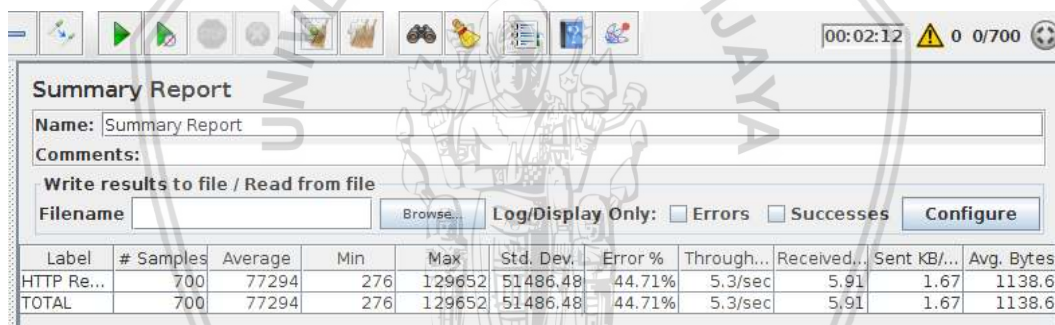
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Through...	Received...	Sent KB/...	Avg. Bytes
HTTP Re...	500	73683	297	127364	50386.84	28.40%	3.8/sec	2.93	1.56	784.1
TOTAL	500	73683	297	127364	50386.84	28.40%	3.8/sec	2.93	1.56	784.1

Gambar 6.28 Summary report test 500 kali

Dari pengujian 500 kali didapatkan hasil bahwa data yang berhasil dikirim yaitu 358 data dengan persentase *error* 28,40% dengan menghabiskan waktu selama 131 detik.

6.2.2.4 Skenario 4 : Melakukan *POST* data sebanyak 700 kali

Pada skenario ini, penguji mencoba melakukan *POST* data sebanyak 700 kali dan didapatkan hasil seperti pada Gambar 6.29 menunjukkan persentase keberhasilan penyimpanan data beserta waktu yang digunakan untuk menyimpan data.



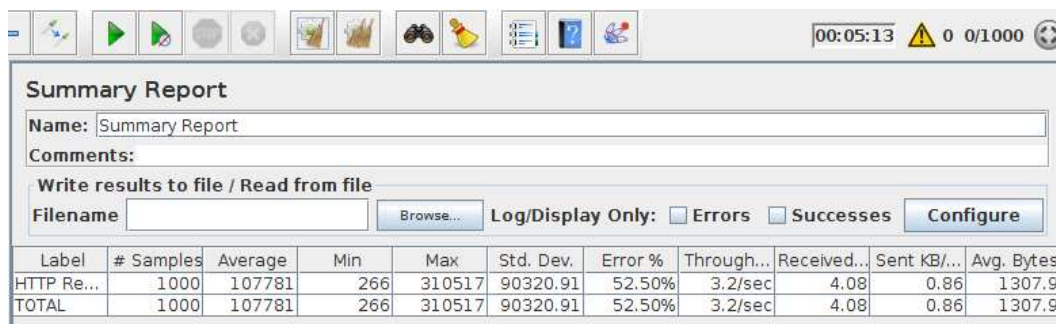
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Through...	Received...	Sent KB/...	Avg. Bytes
HTTP Re...	700	77294	276	129652	51486.48	44.71%	5.3/sec	5.91	1.67	1138.6
TOTAL	700	77294	276	129652	51486.48	44.71%	5.3/sec	5.91	1.67	1138.6

Gambar 6.29 Summary report test 700 kali

Dari pengujian 700 kali didapatkan hasil bahwa data yang berhasil dikirim yaitu 387 data dengan persentase *error* 44,71% dengan menghabiskan waktu selama 132 detik.

6.2.2.5 Skenario 5 : Melakukan *POST* data sebanyak 1000 kali

Pada skenario ini, penguji mencoba melakukan *POST* data sebanyak 1000 kali dan didapatkan hasil seperti pada Gambar 6.30 menunjukkan persentase keberhasilan penyimpanan data beserta waktu yang digunakan untuk menyimpan data.



The screenshot shows the 'Summary Report' window in Apache JMeter. It includes fields for Name, Comments, and options to write results to a file or read from one. Below these is a table of test results.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Through...	Received...	Sent KB/...	Avg. Bytes
HTTP Re...	1000	107781	266	310517	90320.91	52.50%	3.2/sec	4.08	0.86	1307.9
TOTAL	1000	107781	266	310517	90320.91	52.50%	3.2/sec	4.08	0.86	1307.9

Gambar 6.30 Summary report test 1000 kali

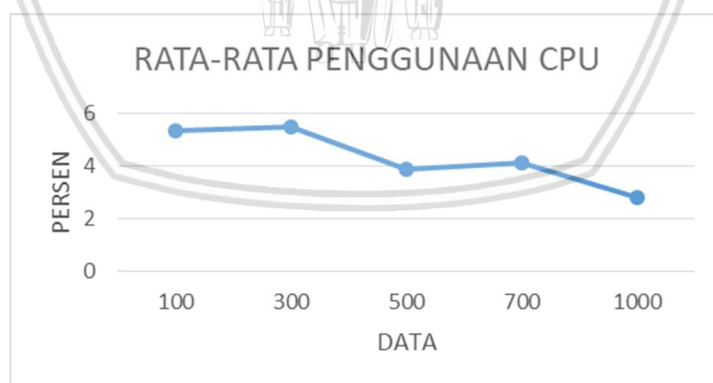
Dari pengujian 1000 request didapatkan hasil bahwa data yang berhasil dikirim yaitu 475 data dengan persentase *error* 52,50% dengan menghabiskan waktu selama 313 detik.

6.3 Pengujian Kinerja API Web Service

Pengujian kinerja merupakan salah satu pengujian yang menunjukkan Kinerja web service ketika sedang melakukan fungsi *post* data dan *get* data. Pengujian ini dilakukan menggunakan Apache JMeter dengan *library plugin manager*.

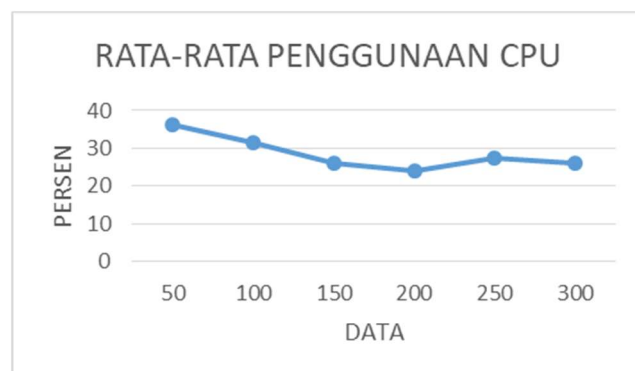
6.3.1 Parameter Penggunaan CPU

Kinerja dari web service bisa dilihat dari persentase penggunaan prosesor yang digunakan ketika menyimpan dan mengambil data pada data *storage*. Gambar 6.31 menunjukkan rata-rata persentase penggunaan prosesor ketika mengirim data ke data *storage*. Dari hasil pengujian yang telah dilakukan rata-rata persentase penggunaan CPU saat melakukan penyimpanan data yaitu 4,32% dengan nilai rata-rata tertinggi yaitu 5,49% dan rata-rata terendah yaitu 2,83%.



Gambar 6.31 Rata-rata CPU usage ketika POST data

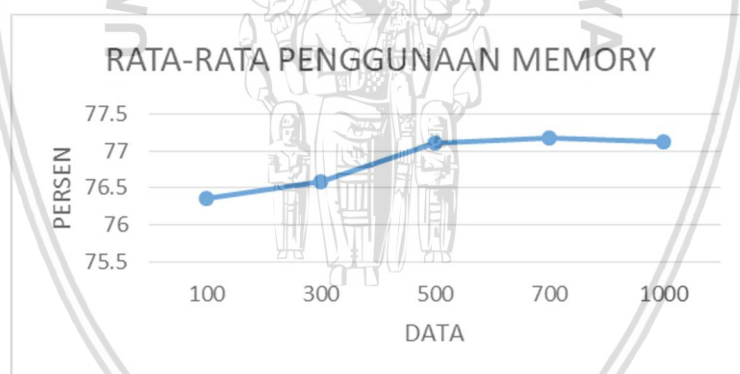
Untuk persentase penggunaan prosesor ketika mengambil data ditunjukkan pada Gambar 6.32. Dari hasil pengujian yang telah dilakukan rata-rata persentase Penggunaan CPU saat melakukan pengambilan data yaitu 28,51% dengan nilai rata-rata tertinggi yaitu 36% dan rata-rata terendah yaitu 25,89%.



Gambar 6.32 Rata-rata CPU usage ketika GET data

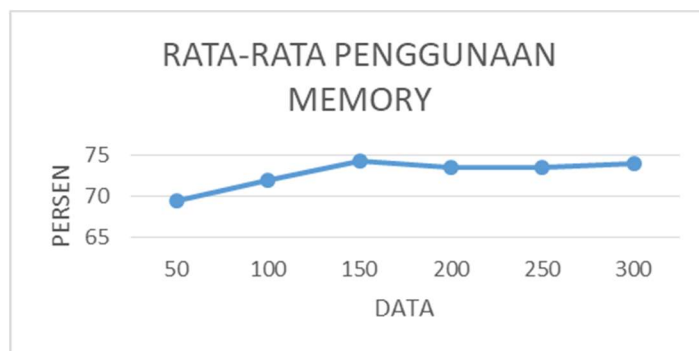
6.3.2 Parameter Penggunaan Memori

Kinerja dari web service bisa dilihat dari persentase penggunaan memori yang digunakan ketika menyimpan dan mengambil data pada data *storage*. Gambar 6.33 menunjukkan rata-rata persentase penggunaan memori ketika menyimpan data ke data *storage*. Dari hasil pengujian yang telah dilakukan rata-rata persentase Memory usage saat melakukan pengiriman data yaitu 76,87% dengan nilai rata-rata tertinggi yaitu 77,17% dan rata-rata terendah yaitu 76,36%.



Gambar 6.33 Rata-rata memory usage ketika POST data

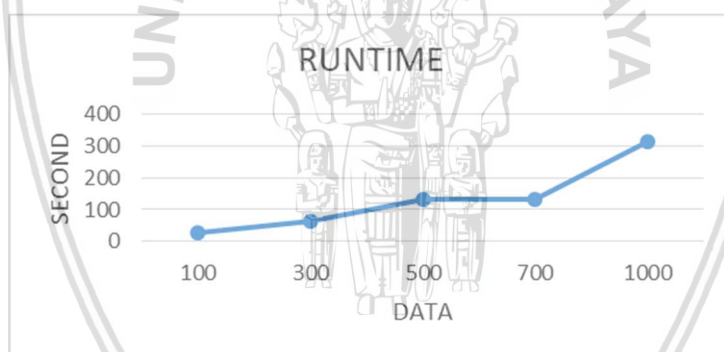
Sedangkan untuk persentase penggunaan memori ketika mengambil data ditunjukkan pada Gambar 6.34. Dari hasil pengujian yang telah dilakukan rata-rata persentase Memory usage saat melakukan pengambilan data yaitu 72,82% dengan nilai rata-rata tertinggi yaitu 74,25% dan rata-rata terendah yaitu 69,52%.



Gambar 6.34 Rata-rata memory usage ketika GET data

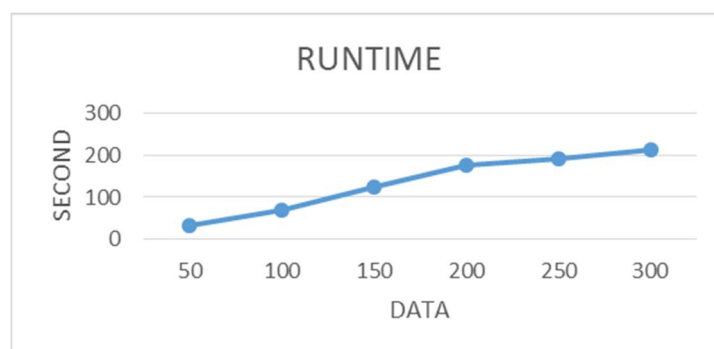
6.3.3 Parameter Runtime

Kinerja dari web *service* bisa dilihat dari waktu yang diperlukan ketika menyimpan dan mengambil data pada data *storage* . Gambar 6.35 menunjukkan waktu yang diperlukan ketika menyimpan data ke data *storage* . Dari hasil pengujian yang telah dilakukan rata-rata waktu yang digunakan saat melakukan penyimpanan data yaitu 133,4 seconds dengan waktu tertinggi yaitu 313 seconds dan waktu terendah yaitu 26 seconds.



Gambar 6.35 Runtime ketika POST data

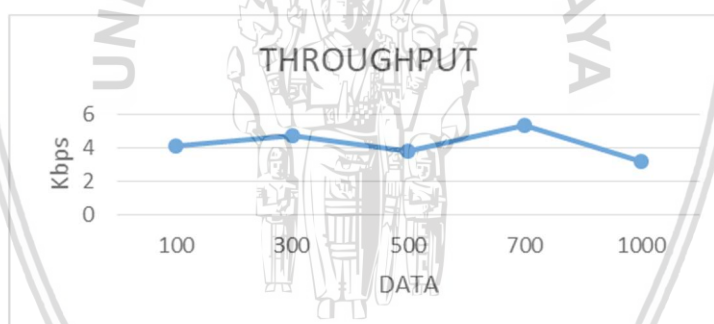
Sedangkan waktu yang digunakan ketika mengambil data ditunjukkan pada Gambar 6.36 . Dari hasil pengujian yang telah dilakukan waktu yang digunakan saat melakukan pengambilan data yaitu 134,33 seconds dengan waktu tertinggi yaitu 213 seconds dan waktu terendah yaitu 33 seconds.



Gambar 6.36 Runtime ketika GET data

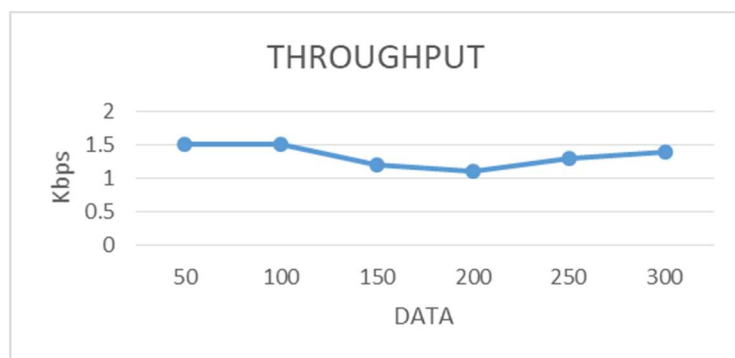
6.3.4 Parameter *Throughput*

Kinerja dari *web service* bisa dilihat dari jumlah operasi yang dieksekusi ketika menyimpan dan mengambil data pada *data storage*. Gambar 6.37 menunjukkan *throughput* dihasilkan ketika menyimpan data ke *data storage*. Dari hasil pengujian yang telah dilakukan rata-rata *throughput* yang dihasilkan saat melakukan pengiriman data yaitu 4,22 KB *per second* dengan *throughput* tertinggi yaitu 5,3 KB *per second* dan *throughput* terendah yaitu 3,2 KB *per second*.



Gambar 6.37 Throughput ketika POST data

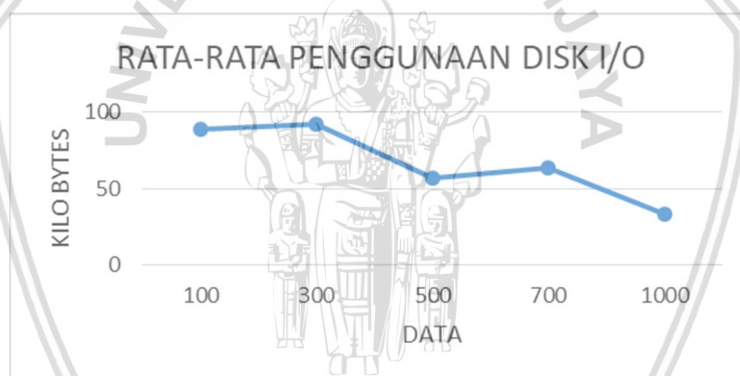
Sedangkan *throughput* yang dihasilkan ketika mengambil data ditunjukkan pada Gambar 6.38. Dari hasil pengujian yang telah dilakukan *throughput* yang dihasilkan saat melakukan pengambilan data yaitu 1,33 KB *per second* dengan *throughput* tertinggi yaitu 1,5 KB *per second* dan *throughput* terendah yaitu 1,1 KB *per second*.



Gambar 6.38 *Throughput* ketika GET data

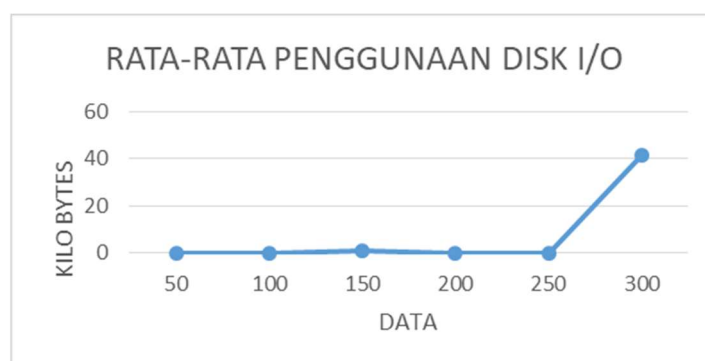
6.3.5 Parameter *Disk I/O*

Kinerja dari web *service* bisa dilihat dari penggunaan disk ketika menyimpan dan mengambil data pada data *storage*. Gambar 6.39 menunjukkan pengguna disk ketika menyimpan data ke data *storage*. Dari hasil pengujian yang telah dilakukan rata-rata kinerja disk saat melakukan penyimpanan data yaitu 67,16 KB dengan kinerja disk tertinggi yaitu 92,62 KB dan kinerja disk terendah yaitu 33,33 KB.



Gambar 6.39 Disk I/O ketika POST data

Sedangkan penggunaan disk ketika mengambil data ditunjukkan pada Gambar 6.40. Dari hasil pengujian yang telah dilakukan kinerja disk yang digunakan saat melakukan pengambilan data yaitu 12,80 KB dengan kinerja disk tertinggi yaitu 41,63 KB dan kinerja disk terendah yaitu 0 KB.



Gambar 6.40 Disk I/O ketika GET data

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan dari perancangan, implementasi, serta pengujian yang dilakukan maka penelitian menghasilkan kesimpulan sebagai berikut :

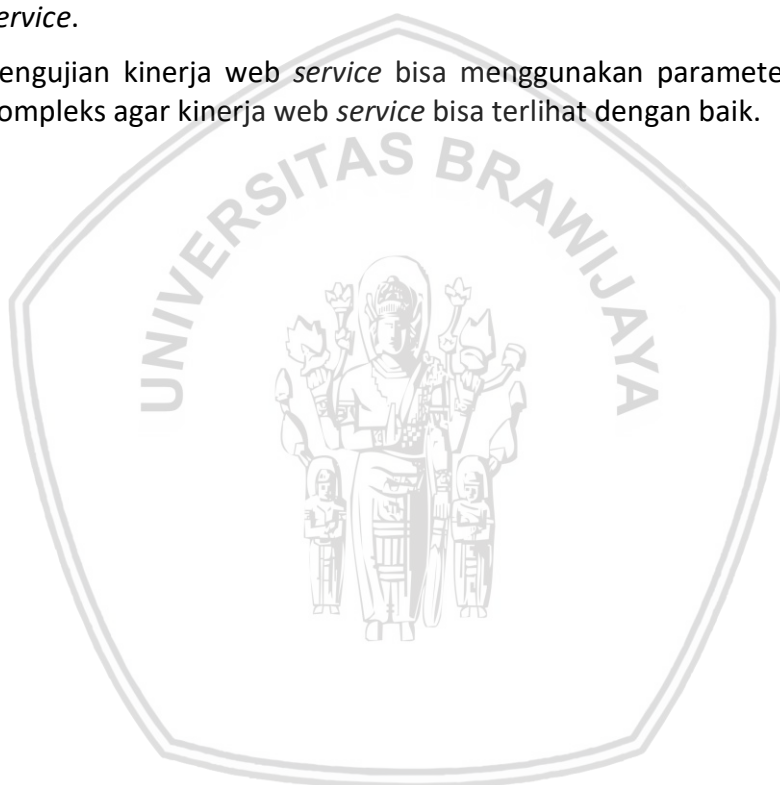
1. API *Restful Web service* dengan konsep *Semantic Interoperability* dapat diimplementasikan dengan cara memperjelas makna dari data tersebut sehingga lebih mudah dipahami. Dalam penelitian ini, *Semantic Interoperability* diterapkan dengan menyimpan meta data yaitu meta data *text* dan meta data gambar serta ukuran data yang telah ditentukan ke dalam data *storage* yang berbeda.
2. Penerapan mekanisme autentikasi dan otorisasi menggunakan JSON Web Token dapat diimplementasikan pada web *service* dengan cara membatasi hak akses web *service*. Hak akses web *service* bisa dibatasi dengan cara pembuatan akun dengan menggunakan *username*, *password*, dan *mac address* dari device yang digunakan. Pembuatan akun diperlukan untuk mendapatkan token sebagai bentuk autentikasi selanjutnya untuk mengakses web *service*.
3. Pada penelitian ini, kinerja API *Restful Web service* dilihat dari Kinerja web *service* ketika digunakan. Parameter yang diujikan menunjukkan nilai yang sangat fluktuatif. Adapun hasil uji kinerja web *service* seperti berikut :
 - a. Rata-rata penggunaan prosesor ketika melakukan fungsi *post* mencapai 10,99% sedangkan ketika melakukan fungsi *get*, penggunaan prosesor bisa mencapai 99%. Nilai-nilai tersebut dipengaruhi oleh banyaknya data yang akan diambil atau disimpan pada data *storage*.
 - b. Rata-rata penggunaan memori ketika melakukan fungsi *post* mencapai 77,52% sedangkan ketika melakukan fungsi *get*, penggunaan prosesor bisa mencapai 77,49%. Nilai-nilai tersebut dipengaruhi oleh banyaknya data yang akan diambil atau disimpan pada data *storage*.
 - c. Waktu yang dibutuhkan ketika melakukan *post* dan *get* sangat dipengaruhi oleh koneksi internet yang digunakan juga banyaknya data yang akan dieksekusi. Waktu yang diperlukan untuk melakukan *get* data yaitu mencapai 313 detik, sedangkan untuk *post* memerlukan waktu mencapai 213 detik untuk mengirim data.
 - d. Throughput yang dihasilkan ketika melakukan *post* data dapat mencapai 5,3 kb/s sedangkan throughput yang dihasilkan ketika *get* data dapat mencapai 1,5 kb/s. Hal ini dapat dipengaruhi oleh koneksi internet yang digunakan dan jumlah data yang akan dieksekusi.
 - e. Rata-rata kinerja disk ketika melakukan fungsi *post* dijalankan bisa mencapai 528 kb dan rata-rata kinerja disk ketika melakukan fungsi *get*

dijalankan mencapai 8136 kb. Nilai-nilai tersebut bergantung pada jumlah data yang akan dieksekusi.

7.2 Saran

Berdasarkan kesimpulan pada penelitian ini, maka peneliti merekomendasikan beberapa hal dalam bentuk saran-saran dengan harapan bisa dilanjutkan untuk penelitian selanjutnya yaitu :

1. Penelitian berikutnya dapat menggunakan meta data yang lebih banyak lagi yang mampu dikelola oleh *web service*.
2. Penyimpanan data yang digunakan lebih banyak lagi dan sesuai dengan jumlah meta data yang digunakan dalam pertukaran data pada *web service*.
3. Pengujian kinerja *web service* bisa menggunakan parameter yang lebih kompleks agar kinerja *web service* bisa terlihat dengan baik.



DAFTAR PUSTAKA

- Anwari, H., 2017. Pengembangan IoT *Middleware* Berbasis Event-Based Dengan Protokol Komunikasi COAP, MQTT, dan Websocket.
- Arganata, G., 2017. Pengembangan Sistem Penyimpanan Data Berbasis MongoDB dan GridFS.
- Atzori, L., Iera, A. & Morabito, G., 2010. *The Internet of Things : A Survey*. Elsevier, pp. 2787 - 2805.
- European Research Cluster on the Internet of Things, 2015. *IoT Semantic Interoperability: Research Challenges, Best Practices, Recommendations and Next Steps*. 1.5 penyunt. s.l.:European Commission Information Society and Media.
- Grgic, K., Hedi, I. & S. H., 2016. A Web-Based IoT Solution for Monitoring Data Using MQTT Protocol. *IEEE*, pp. 249-253.
- Guinard, D., Ion, I. & Mayer, S., 2011. In Search of an Internet of Things Service Architecture: REST or WS-*? A Developers' Perspective. *Proc. the Eight International ICST Conference on Mobile and Ubiquitous System*.
- Haekal, M. & Eliyani, 2016. Token-Based Authentication Using JSON Web Token on SIKASIR RESTful Web Service. *IEEE*.
- Laine, M., 2011. *RESTful Web Service for The Internet of Things*.
- Li, X., Yu, Y., Wang, H. & Qian, C., 2017. An IoT Data Communication Framework for Authenticity and Integrity. *IEEE International Conference on Internet-of-Things Design and Implementation*, pp. 159-170.
- Online, K., 2016. *KBBI Daring*. [Online] Available at: <https://kbbi.kemdikbud.go.id> [Diakses 23 March 2018].
- Putra, A. W. P., 2017. Implementasi Autentikasi JSON Web Token (JWT) Sebagai Mekanisme Autentikasi Protokol Message Queuing Telemetry Transport (MQTT) Pada Perangkat NodeMCU. *J-PTIIK*.
- Rodriguez, A., 2015. *RESTful Web Services: The Basics*. s.l.:IBM developer works.